The Word Processor
for the Commodore 64 and VIC-20

Charles Brannon

A **COMPUTE! Books** Publication

$9.95

SpeedScript

The Word Processor for the Commodore 64 and VIC-20

Charles Brannon

COMPUTE!™ Publications, Inc. abc
One of the ABC Publishing Companies

Greensboro, North Carolina

The following articles were originally published in *COMPUTE!* magazine, copyright 1985, COMPUTE! Publications, Inc.: "SpeedScript 3.0: All Machine Language Word Processor For Commodore 64" (March), "SpeedScript 3.0: All Machine Language Word Processor For Expanded VIC-20" (April), and "ScriptSave" (May).

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-942386-94-9

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is one of the ABC Publishing Companies and is not associated with any manufacturer of personal computers. Commodore 64 and VIC-20 are trademarks of Commodore Electronics Limited.

Contents

Foreword

SpeedScript is the most popular program ever published by COMPUTE! Publications. Ever since its first appearance in the January 1984 issue of *COMPUTE!'s Gazette*, the letters have been pouring in. People wanted to know more about the program and word processing, and they had countless suggestions about how to make *SpeedScript* better.

The result is *SpeedScript 3.1*, an even more powerful word processor. Enhanced with additional commands and features, this machine language word processor gives you all the things you expect from a commercial software package. You can write, edit, format, and print anything from memos to novels on your Commodore 64 or Expanded VIC-20. With a few keystrokes you can change the color of the screen and its text to whatever combination best suits you.

It's easy to add or delete words, letters, even whole paragraphs. You can search through an entire document and find every occurrence of a particular word or phrase, then replace it with something new. Of course, when you finish writing, you can save your work to tape or disk.

The ability to quickly change the appearance of a printed document is one of the things that makes word processing so efficient. *SpeedScript* lets you alter the margins, page length, spacing, page numbers, page width, and set up headers and footers at the top and bottom of the paper. Once you've formatted your document, you can print it out.

There are enough print features to make even the most demanding writer happy. With *SpeedScript*, you can start printing at any page, force the printer to create a new page at any time, even make it wait while you put in another sheet of paper. Underlining and centering are simple. If you want to get fancy, you can use your printer's codes to create graphics symbols or logos. And if you're writing something *really* long, perhaps a novel or term paper, *SpeedScript* lets you link any number of files so that they print out as one continuous document.

In addition to the *SpeedScript* programs for the 64 and VIC-20, you'll find complete documentation, a keyboard overlay, and two quick-reference cards included in this book. *SpeedScript*'s source code has also been included for your

examination. By studying it, you'll see exactly how the program is put together. An additional program, "ScriptSave," can be added to *SpeedScript* to automatically save your files every ten minutes. This can be a lifesaver if there's a power failure. "The Machine Language Editor: MLX" makes typing in the program easier. MLX almost guarantees that you'll have an error-free copy of the program the first time you type it in. If you prefer to purchase a copy of *SpeedScript* on disk rather than type it in, just use the convenient coupon in the back, or call toll-free 1-800-334-0868.

Chapter 1
Using *SpeedScript*

SpeedScript 3.1
All Machine Language Word Processor for the Commodore 64 and VIC-20

Since its introduction in the January 1984 issue of COMPUTE!'s Gazette, SpeedScript *has been the most popular program ever published by COMPUTE! Publications. Written entirely in machine language,* SpeedScript *contains nearly every command and convenience you'd expect from a quality word processor.*

The version of SpeedScript *in this book, version 3.1, incorporates a year's worth of improvements, readers' suggestions, and additional debugging.*

This book contains all the documentation and listings you need to enter and use SpeedScript *on the Commodore 64 or VIC-20, with 8K or more expansion memory.*

SpeedScript 3.1, though compact in size (6K), has many features found on commercial word processors. *SpeedScript* is also very easy to learn and use. You can start writing with it the first time you use it. You type in everything first; preview and make corrections on the screen; insert and delete words, sentences, and paragraphs; and then print out an error-free draft, letting *SpeedScript* take care of things like margins, centering, headers, and footers.

SpeedScript is a writing tool. It won't necessarily make you a better writer, but you may become a better writer once the tedium of retyping and erasing is replaced by the flexibility of a word processor. Words are no longer frozen in place by ink; they become free-floating entities. You no longer think about typewriting; you can stand back and work directly with words and ideas. The distinction between rough and final drafts becomes blurred as you perfect your writing while you write it.

Typing In *SpeedScript*

SpeedScript is one of the longest machine language programs COMPUTE! has ever published, but the "MLX" machine language entry system helps you type it right the first time. MLX

also lets you type *SpeedScript* in more than one sitting. Unfortunately, if you have an earlier version of *SpeedScript*, you cannot just make certain changes to bring it up to version 3.1. You have to type it in from scratch. Although this might seem daunting, I'm sure you'll find it worthwhile.

If you prefer not to type in *SpeedScript*, you can purchase a copy on disk from COMPUTE! Publications by either using the coupon in the back of this book or calling toll-free 1-800-334-0868.

Using MLX

MLX makes it possible for you to type in a long machine language program correctly. It can detect most errors people make when entering numbers. See the MLX article in Chapter 2.

Before you begin typing *SpeedScript* (or begin a subsequent session of typing if you enter *SpeedScript* in more than one sitting), you must enter certain POKEs *before* you load and run the MLX program (be sure you use the proper POKEs for your computer). These POKEs are essential to protect *SpeedScript* from BASIC while you are typing it in. Again, these POKEs should be performed *before* you load MLX, but are not necessary to run the finished *SpeedScript* program:

Commodore 64:

POKE 44,33:POKE 8448,0:NEW

VIC-20:

POKE 44,42:POKE 10752,0:NEW

Now load and run the version of MLX for your computer (VIC-20 users must have at least 8K memory expansion to run VIC-20 MLX). When you run MLX, the first thing you must enter is the starting and ending address of *SpeedScript*.

If you are using a Commodore 64, enter:

Starting Address? 2049
Ending Address? 8204

If you are using a VIC-20, enter:

Starting Address? 4609
Ending Address? 10482

You will then see the first prompt, the number 2049, on the Commodore 64, or the number 4609, on the VIC-20, followed by a colon. Type in each three-digit number shown in

the listing. You do not need to press the comma shown in the listing; MLX types the comma automatically.

The last number you enter in a line is a *checksum*. It represents the values of the other numbers in the line summed together. If you make a mistake while entering the line, the checksum calculated by MLX should not match that of the listing, and you will have to retype the line. MLX is not foolproof, though. It's possible to fool the checksum by exchanging the position of the three-digit numbers. Also, an error in one number can be offset by an error in another (just as 3 + 4 + 7 = 1 + 4 + 9). Keep this in mind. MLX will help catch your errors, but you still must be very careful.

Typing in Multiple Sittings

If you want to stop typing the listing at some point and pick up later, press SHIFT-S and follow the screen prompts. Be sure you have a tape or disk ready with room to store a 6K program (about 25 disk blocks). Remember to note the line number of the last complete line you typed in. While entering *SpeedScript*, you should use a different filename for each partially complete version you save. If you use an existing filename, MLX will overwrite the existing file as it saves the newest version. By using different names, you can preserve portions of your work should problems arise. Once you have a complete working version of *SpeedScript*, the partial versions can be erased.

When you are ready to continue typing, enter the proper POKEs mentioned above, load MLX, answer the starting and ending address prompts with the values shown above, and then press SHIFT-L. (Always use the starting and ending addresses shown, regardless of where you stopped typing.) MLX asks for the filename you gave to the partially typed program. After the LOAD is complete, press SHIFT-N and tell MLX the line number you stopped at. (Be sure the line number you enter matches one of the line numbers in the listing.) Now continue typing as before. When you finish all typing, MLX automatically prompts you to save the program.

At this point MLX has saved a program file on tape or disk. If you load it and list it, you'll see that it looks like a normal one-line BASIC program, with a line number and a SYS command. The machine language program that is *SpeedScript* starts in memory just after the SYS command. The

simulated BASIC line is included so that you can load *SpeedScript* like any BASIC program and enter RUN to start it. You don't need to add the ,1 like you do with many machine language programs. Just LOAD "SPEEDSCRIPT" (or whatever filename you called it) for tape or LOAD "SPEEDSCRIPT",8 for disk, then enter RUN. Once *SpeedScript* is in memory, you can save it from BASIC like any BASIC program. To exit to BASIC while *SpeedScript* is running, tap the RESTORE key on the Commodore 64 or press RUN/STOP–RESTORE on the VIC-20.

Before using *SpeedScript*, you should generally unplug all cartridges and expanders such as *Simons' BASIC* or the *Super Expander*. (On the VIC-20, you must have a memory expansion cartridge plugged in that provides at least an additional 8K, although *SpeedScript* can take advantage of up to 24K of memory expansion.) *SpeedScript* cannot take advantage of any custom hardware configurations except those that do not interfere with normal operations.

Entering Text

When you run *SpeedScript*, the screen colors change to dark gray on light gray on the Commodore 64, and black on white on the VIC-20. The first screen line on the Commodore 64 (or the first two lines on the VIC-20) is black with white letters. This *command line* is used to communicate with *SpeedScript*. *SpeedScript* presents all messages here. The remaining lines of the screen are used to enter, edit, and display your document. The *cursor* shows where the next character you type will appear on the screen. *SpeedScript* lets you move the cursor anywhere within your document, making it easy to find and correct errors.

To begin using *SpeedScript*, just start typing. When the cursor reaches the right edge of the screen, it automatically jumps to the beginning of the next line, just as in BASIC. But unlike BASIC, *SpeedScript* never splits words at the right edge of the screen. If a word you're typing won't fit at the end of one line, it's instantly moved to the next line. This feature, called *word-wrap*, or sometimes *parsing*, makes it much easier to read your text on the screen. Even if you make numerous editing changes, *SpeedScript* reformats the screen and rewraps all words.

Scrolling and Screen Formatting

When you finish typing on the last screen line, *SpeedScript* automatically scrolls the text upward to make room for a new line at the bottom. This is similar to the way BASIC works, but with one exception: The screen can scroll both up *and* down. Imagine the screen as a 24-line (Commodore 64) or 21-line (VIC-20) window on a long continuous document.

The Commodore 64 has more than 43K of text space available in memory, room enough for 20–40 printed pages of text. On the VIC-20, there's room for 3072 characters of text with an 8K expander or up to 19,456 with a 24K expander.

To check at any time how much space is left, press **CTRL-=** (hold down the CTRL key while pressing the = key). The number which appears in the command line indicates how much room remains for characters of text.

If you're used to a typewriter, you'll have to unlearn some habits. First, since the screen is only 40 (Commodore 64) or 22 (VIC-20) columns wide, and most printers have 80-column carriages, it doesn't make sense to press RETURN at the end of each line as you do on a typewriter. *SpeedScript*'s word-wrap takes care of this automatically. Press RETURN only when you want to force a carriage return to end a paragraph or to limit the length of a line. To permit you to see these forced carriage returns, they appear on the screen as a left-pointing arrow. (This is called a *return mark* in this book.)

When you print your document, *SpeedScript* automatically formats your text to fit the width of the paper. Don't manually space over for a left margin or try to center a line yourself as you would on a typewriter. *SpeedScript*'s printing routine automatically takes care of all margins and lets you customize the margin settings. Also, don't worry about where a printed page will end. When printing, *SpeedScript* automatically fits your text onto separate pages and can even put short phrases and page numbers at the top or bottom of each page if you want.

Like all good word processors, *SpeedScript* has a wide selection of editing and convenience features. You can move the cursor a single space in either direction, or skip to the next or previous word, sentence, or paragraph. You can also move the cursor to the top of the screen, the top of the document, or the end of the document. The INST/DEL key is used to insert a single space or delete a single character. Other features let

you erase a word, sentence, or paragraph, and move or copy sentences, words, and paragraphs to other places in your document. Using Search and Replace, you can find any phrase and even automatically change one phrase to another throughout the entire document.

You can save your text on tape or disk, then load it later for additions and corrections. You can transpose (exchange) two characters, change the screen and text colors, send disk commands, read the disk error channel, and automatically tab over five spaces for paragraph indents. You don't need to learn all these commands right away, but you'll be glad they're available as you become more comfortable with word processing.

Using the Keyboard

Most of these features are accessed with control-key commands—you hold down CTRL while pressing another key. In this book, control-key commands are abbreviated **CTRL-*x*** (where *x* is the key you press in combination with CTRL). An example is the **CTRL-=** mentioned above to check on free memory. **CTRL-E** means hold down CTRL and press E. Sometimes you have to hold down both SHIFT and CTRL as you type the command key, as in SHIFT–CTRL-H. Other keys are referenced by name or function, such as back arrow for the left-pointing arrow in the top-left corner of the keyboard, pound sign for the British pound sign (£), CLR/HOME for the Home Cursor key, SHIFT–CLR/HOME for the Clear Screen key, f1 for special function key 1, and up arrow for the upward-pointing arrow to the left of the RESTORE key. See Appendix D for a complete quick-reference chart of all keyboard commands or Figure 1-1 (below) for a keyboard map.

Some keys let you move the cursor to different places in the document to make corrections or scroll text into view. *SpeedScript* uses a unique method of cursor movement that is related to writing, not programming. Programmers work with lines of text and need to move the cursor up and down a line or left and right across a line. *SpeedScript,* however, is oriented for writers. You aren't working with lines of text, but with a continuous document.

Therefore, *SpeedScript* moves the cursor by character, word, sentence, or paragraph. *SpeedScript* defines a word as any sequence of characters preceded or followed by a space. A

sentence is any sequence of characters ending with a period, exclamation point, question mark, or return mark. And a paragraph is defined as any sequence of characters ending in a return mark. (Again, a return mark appears on the screen as a left-pointing arrow.)

Here's how to control the cursor:

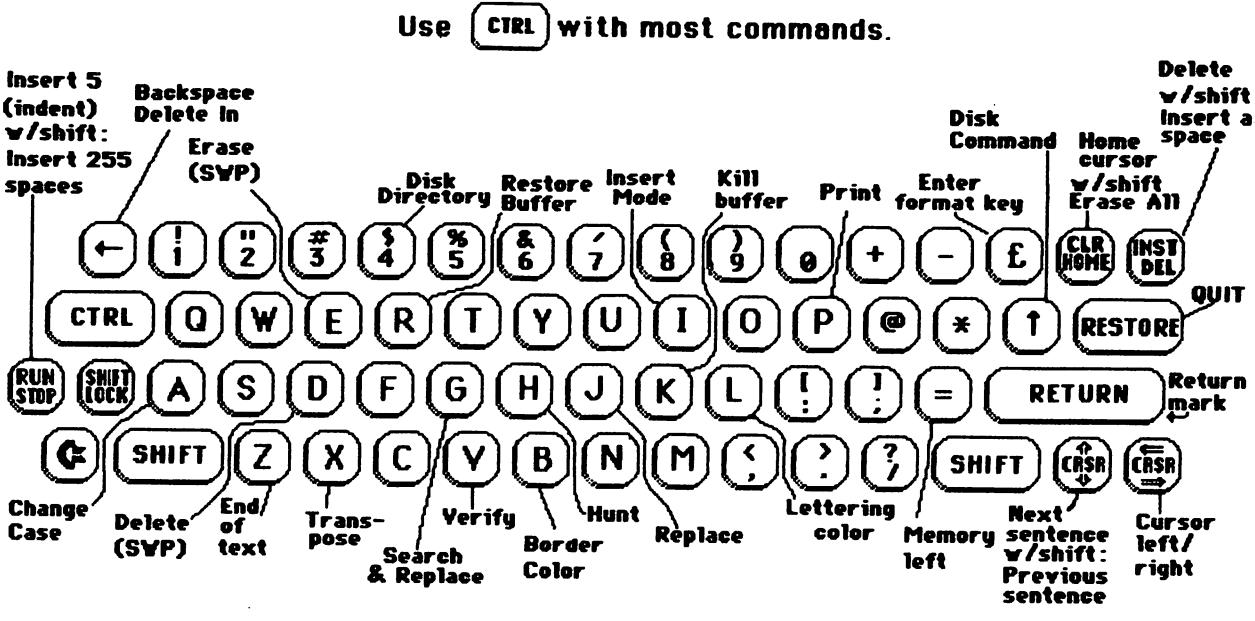
- The **left/right cursor key** works as usual; pressing this key by itself moves the cursor right (forward) one space, and pressing it with SHIFT moves the cursor left (backward) one space.
- The **up/down cursor key** moves the cursor forward to the beginning of the next sentence. Pressing it with SHIFT moves the cursor backward to the beginning of the previous sentence.
- The **f1 special function key** moves the cursor forward to the beginning of the next word. The **f2 key** (hold down SHIFT and press f1) moves the cursor backward to the beginning of the previous word.
- The **f3 special function key** moves the cursor forward to the beginning of the next sentence (just like the up/down cursor key). The **f4 key** (hold down SHIFT and press f3) moves the cursor backward to the beginning of the previous sentence (just like pressing SHIFT and the up/down cursor key).
- The **f5 special function key** moves the cursor forward to the beginning of the next paragraph. The **f6 key** (hold down SHIFT and press f5) moves the cursor backward to the beginning of the previous paragraph.
- The **CLR/HOME key,** pressed once by itself, moves the cursor to the top of the screen without scrolling. Pressed twice, it moves the cursor to the beginning of the document.
- **CTRL-Z** moves the cursor to the bottom of the document.

Correcting Your Typing

One strength of a word processor is that you need never have mistakes in your printed document. Since you've typed everything before you print it, you have plenty of opportunities to proofread and correct your work. The easiest way to correct something is just to type over it, but there are other ways, too.

Sometimes you'll have to insert some characters to make a correction. Maybe you accidentally dropped a letter, typing *hngry* instead of *hungry.* When you change the length of a

Figure 1-1. *SpeedScript* Keyboard Map



word, you need to push over everything to the right of the word to make room for the insertion. Use **SHIFT-INST/DEL** to open up a single space, just as in BASIC. Merely position the cursor at the point where you want to insert a space, and press SHIFT-INST/DEL.

Insert Modes

It can be tedious to use the SHIFT-INST/DEL key to open up enough space for a whole sentence or paragraph. For convenience, *SpeedScript* has an insert mode that automatically inserts space for each character you type. In this mode, you can't type over characters; everything is inserted at the cursor position. To enter insert mode, press **CTRL-I.** To cancel insert mode, press CTRL-I again (a command key that turns something on and off is called a *toggle*). To let you know you're in insert mode, the normally black command line at the top of the screen turns blue.

Insert mode is the easiest way to insert text, but it can become too slow when working with a very long document because it must move *all* the text following the cursor position. Although *SpeedScript* uses turbocharged memory-move routines, the 6502/6510 microprocessor can go only so fast. So *SpeedScript* has even more ways to insert blocks of text.

One way is to use the **RUN/STOP** key. It is programmed in *SpeedScript* to act as a five-space margin indent. To end a paragraph and start another, press RETURN twice and press RUN/STOP. Alternatively, you can press **SHIFT-RETURN,** which does this automatically. You can use RUN/STOP to open up more space than SHIFT-INST/DEL. No matter how much space you want to insert, each insertion takes the same amount of time. So the RUN/STOP key can insert five spaces five times faster than pressing SHIFT-INST/DEL five times.

There's an even better way, though. Press **SHIFT-RUN/STOP** to insert 255 spaces. This is enough room for a sentence or two. You can press it several times to open up as much space as you need. And SHIFT-RUN/STOP is *fast.* (You don't want to be in insert mode when you use this trick; that would defeat its purpose.)

Since the INST/DEL key is also slow when working with large documents (it, too, must move all text following the cursor), you may prefer to use the back-arrow (←) key to backspace. The **back-arrow** key by itself moves the cursor left one

space and blanks out that position. It's more like a backspace than a delete.

After you're done inserting with these methods, there will probably be some inserted spaces left over that you didn't use. Just press **SHIFT-CTRL-back arrow**. This instantly deletes all extra spaces between the cursor and the start of following text. SHIFT-CTRL-back arrow is also generally useful whenever you want to delete a bunch of spaces.

Erasing Text

Inserting and retyping are not the only kinds of corrections you'll need to make. Part of writing is separating the wheat from the chaff. On a typewriter, you pull out the paper, ball it up, and dunk it in the trash can. *SpeedScript* lets you be more selective.

Press the **INST/DEL** key by itself to erase the character to the left of the cursor. All the following text is pulled back to fill the vacant space.

Press **CTRL-back arrow** to delete the character on which the cursor is sitting. Again, all the following text is moved toward the cursor to fill the empty space.

These keys are fine for minor deletions, but it could take all day to delete a whole paragraph this way. So *SpeedScript* has two commands that can delete an entire word, sentence, or paragraph at a time. **CTRL-E** erases text *after* (to the right of) the cursor position, and **CTRL-D** deletes text *behind* (to the left of) the cursor.

To use the **CTRL-E** erase mode, first place the cursor at the beginning of the word, sentence, or paragraph you want to erase. Then press CTRL-E. The command line shows the message "Erase (S,W,P): RETURN to exit." Press S to erase a sentence, W for a word, or P for a paragraph. Each time you press one of these letters, the text is quickly erased. You can keep pressing S, W, or P until you've erased all the text you wish. Then press RETURN to exit the erase mode.

The **CTRL-D** delete mode works similarly, but deletes only one word, sentence, or paragraph at a time. First, place the cursor after the word, sentence, or paragraph you want to delete. Then press CTRL-D. Next, press S, W, or P for sentence, word, or paragraph. The text is immediately deleted and you return to editing. You don't need to press RETURN to exit

the CTRL-D delete mode unless you pressed this key by mistake. *(In general, you can escape from any command in* SpeedScript *by simply pressing RETURN.)* CTRL-D is most convenient when the cursor is already past what you've been typing.

The Text Buffer

When you erase or delete with CTRL-E and CTRL-D, the text isn't lost forever. *SpeedScript* remembers what you've removed by storing deletions in a separate area of memory called a *buffer.* The buffer is a fail-safe device. If you erase too much or change your mind, just press **CTRL-R** to restore the deletion. However, be aware that *SpeedScript* remembers only the last erase or delete you performed.

Another, more powerful use of this buffer is to move or copy sections of text. To move some text from one location in your document to another, first erase or delete it with CTRL-E or CTRL-D. Then move the cursor to where you want the text to appear and press **CTRL-R.** CTRL-R instantly inserts the contents of the buffer at the cursor position. If you want to copy some text from one part of your document to another, just erase or delete it with CTRL-E or CTRL-D, restore it at the original position with CTRL-R, then move the cursor elsewhere and press CTRL-R to restore it again. You can retrieve the buffer with CTRL-R as many times as you like.

Important: The CTRL-E erase mode lets you erase up to the maximum size of the buffer (12K, or over 12,000 characters on the Commodore 64; or 1K, or 1024 characters on the VIC-20), and CTRL-E also removes the previous contents of the buffer. Keep this in mind if there's something in the buffer you'd rather keep. If you don't want the buffer to be erased, press **SHIFT-CTRL-E.** This preserves the buffer contents and adds newly erased text to the buffer.

Now you can see why CTRL-D lets you delete only a single sentence, word, or paragraph at a time. If it didn't, the deleted text would be added to the end of the buffer, and when you pressed CTRL-R to retrieve the buffer, the deleted text would be out of order (since CTRL-D deletes backward).

If you ever need to erase the contents of the buffer, press **CTRL-K** (remember *kill buffer*).

It's relatively easy to move blocks of text between documents. Using the buffer, you can load one document, erase

some text into the buffer, load another document, then insert the buffer. You can also use the buffer to save an often-used word or phrase, then repeat it whenever you need it.

The Wastebasket Command

If you want to start a new document or simply obliterate all your text, press **SHIFT-CLR/HOME**. *SpeedScript* asks, "ERASE ALL TEXT: Are you sure? (Y/N)." This is your last chance. If you *don't* want to erase the entire document, press N or any other key. Press Y to perform the irreversible deed. There is no way to recover text wiped out with Erase All.

Pressing just RESTORE on the Commodore 64 brings up the message "Exit *SpeedScript*: Are you sure? (Y/N)." If you press Y for yes, you exit to BASIC (if you press N or any other key at the prompt, you return to editing text with no harm done). Pressing RUN/STOP–RESTORE on the VIC-20 will put you back in BASIC. Once in BASIC you'll still have one chance to reenter *SpeedScript* without losing your text—simply enter RUN (but your chances decrease if you execute other commands in BASIC).

Search and Replace

Here's another feature only a computer can bring to writing. *SpeedScript* has a Hunt command that searches through your document to find a selected word or phrase. A Replace option lets you automatically change one word to another throughout the document. Since on the 64, CTRL-S is synonymous with the CLR/HOME key (try it), and since *SpeedScript* already uses CTRL-R, I have to resort to command keys which are slightly less than mnemonic for these functions.

SHIFT-CTRL-H activates the Hunt feature, **SHIFT-CTRL-J** (J is used because it's next to the H) lets you selectively hunt and replace, and **CTRL-G** (Global) is for automatically searching and replacing.

Searching for something is a two-step process. First, you need to tell *SpeedScript* what to search for, then you trigger the actual search. Press SHIFT–CTRL-H. The command line says "Hunt for:". Type in what you'd like to search for, the *search phrase*, up to 29 characters on the Commodore 64 and 34 on the VIC-20. *SpeedScript* remembers the search phrase until you

change it. (Incidentally, when you are typing on the command line, the only editing key that works is the INST/DEL key for backing up. *SpeedScript* does not let you enter control codes or cursor controls when you type in the command line, and you can type no more than one screen line.) Press RETURN when you've finished typing. If you press RETURN alone without typing anything, the Hunt command is canceled.

When you are ready to search, press **CTRL-H.** *SpeedScript* looks for the next occurrence of the search phrase *starting from the current cursor position.* If you want to hunt through the entire document, press CLR/HOME twice to move the cursor to the very top before beginning the search. Each time you press CTRL-H, *SpeedScript* looks for the next occurrence of the search phrase and places the cursor at the start of the phrase. If the search fails, you'll see the message "Not Found."

CTRL-J (Replace) works together with CTRL-H. After you've specified the search phrase with SHIFT–CTRL-H, press SHIFT–CTRL-J to select the replace phrase. *SpeedScript* also remembers this replace phrase until you change it. (You can press RETURN alone at the "Replace with:" prompt to select a *null* replace phrase. When you hunt and replace, this deletes the located phrase.) To manually search and replace, start by pressing CTRL-H. After *SpeedScript* finds the search phrase, press CTRL-J if you want to replace the phrase. If you don't want to replace the phrase, don't press CTRL-J. You are not in a special search and replace mode. You're free to continue writing at any time.

CTRL-G links CTRL-H and CTRL-J together. It first asks "Hunt for:", then "Replace with:", then automatically searches and replaces throughout the document starting at the cursor position.

A few hints and cautions: First, realize that if you use *the* as the search phrase, *SpeedScript* dutifully finds the embedded *the* in words like *therefore* and *heathen.* If you changed all occurrences of *the* to *cow,* these words would become *cowrefore* and *heacown.* If you want to find or replace a single word, include a space as the first character of the word, since almost all words are preceded by a space. Naturally, if you are replacing, you need to include the space in the replace phrase, too.

Also, *SpeedScript* distinguishes between uppercase and lowercase. The word *Meldids* does not match with *meldids. SpeedScript* will not find a capitalized word unless you capitalize it in the search phrase. To cover all bases, you will sometimes need to make two passes when replacing a word. Keep these things in mind when using CTRL-G, since you don't have a chance to stop an out-of-control search and replace.

Storing Your Document

Another advantage of word processing is that you can store your writing on tape or disk. A Commodore disk, with 170K of storage space, can store 80–150 pages of text as several document files. Tapes also have great storage capacity, but they're slower, and it's harder to locate one of several documents on a cassette. However, *SpeedScript* can be used with tape, making it possible to set up an extremely economical word processing system.

SpeedScript can also be used as a simple database manager. Type in the information you need, then store it as a *SpeedScript* document. The search feature lets you quickly find information, especially if you use graphics characters to flag key lines. You can search for the graphics characters and quickly skip from field to field.

It's easy to store a document. First, make sure the cassette or disk drive is plugged in and functioning. Insert the tape and rewind it, or insert a formatted (NEWed) disk into the drive. Press **f8** (SHIFT-f7). You'll see the prompt "Save:". Type in a filename for your document. A filename can be up to 16 characters long and can include almost any characters, but do not use question marks or asterisks. You cannot use the same name for two different documents on a single disk. To replace a document already on disk using the same filename, precede your filename with the characters **@0:** or **@:**. You can also precede the filename with either **0:** or **1:** if you use a dual disk drive. *SpeedScript* cannot access a second disk drive with a device number of 9.

After entering the filename, answer the prompt "Tape or Disk" by pressing either the **T** or **D** key. You can cancel the SAVE command by pressing RETURN without typing anything else at either the "Save:" or "Tape or Disk?" prompt.

After you press T for tape, press RECORD and PLAY simultaneously on the cassette drive. *SpeedScript* begins saving

your document. If you press D for disk, and the disk is formatted and has room, your file is stored relatively quickly. After the SAVE, *SpeedScript* reports "No errors" if all is well, or reads and reports the disk error message if not.

It is not possible to detect errors during a tape SAVE, so if you want peace of mind, use the Verify command. Rewind the tape, press **CTRL-V**, then type the filename. Press T for tape, then press PLAY on the recorder. *SpeedScript* compares the file on tape with that in memory, and reports "No errors" if the verify succeeds, or "Verify Error" if not. You can also verify disk files.

Loading a Document

To recall a previously saved document, press **f7**. Answer the "Load:" prompt with the filename. (Disk users can use CTRL-4—explained later in this chapter—to check the disk directory for the desired filename.) Insert the tape or disk, rewind the tape, then answer T or D. Press PLAY on tape. *SpeedScript* loads the file and should display "No errors." Otherwise, *SpeedScript* reads the error channel of the disk drive or simply reports "Load error" for tape.

The position of the cursor is important before loading a file. *SpeedScript* starts loading at the cursor position, so be sure to press CLR/HOME twice or SHIFT–CLR/HOME (Erase All) to move the cursor to the start of text space, unless you want to merge two documents. When you press f7 to load, the command line turns green to warn you if the cursor is not at the top of the text space.

To merge two or more files, simply load the first file, press CTRL-Z to move the cursor to the end of the document, and then load the file you want to merge. Do not place the cursor somewhere in the middle of your document before loading. A LOAD does not insert the characters coming in from tape or disk into your old text, but overwrites all existing text after the cursor position. The last character loaded becomes the new end-of-text marker, and you cannot access any of your old text that may appear after this marker.

Commodore 64 File Compatibility

SpeedScript documents are stored as program files (a PRG type on disk). Naturally, you can't load and run a *SpeedScript* file from BASIC. Program files on tape are more reliable than data

files. The characters are stored in their screen code (POKE) equivalents. Several commercial word processors store text similarly, including *WordPro 3+*, *PaperClip*, and *EasyScript*. As a matter of fact, two commercial spelling checkers designed for *WordPro* also work with *SpeedScript*: *SpellRight Plus* (from Professional Software) and *SpellPro 64* (from Pro-Line).

Program 2-6 is a *SpeedScript* file conversion utility for the Commodore 64. It translates *SpeedScript* screen-code program files into either Commodore ASCII or true ASCII. These translated files are stored in SEQuential format, the file type used in most file-processing applications. The file converter program can also translate a Commodore ASCII sequential file into a screen-code *SpeedScript* program file. You can use the file converter to translate a database into a *SpeedScript* file (or vice versa), and you can convert *SpeedScript* files to true ASCII and use a modem program to upload them to another computer.

Disk Commands

Sometimes you forget the name of a file or need to scratch or rename a file. *SpeedScript* gives you full control over the disk drive. To view the disk directory, press **CTRL-4**. The directory will be displayed on the screen without affecting the text in memory. You can press any key to pause scrolling. Afterward, press RETURN to switch back to your text. All the other disk commands are also accessible. Just press **CTRL-↑** (up arrow), then type in a 1541 disk command. You don't need to type PRINT#15 or any quotes as you do in BASIC, just the actual command. If you press RETURN without typing a disk command, *SpeedScript* displays the disk status. It also displays the status after completing a disk command. Here is a quick summary of disk commands:

n:*disk name*,ID This formats (NEWs) a disk. You must format a new disk before using it for the first time. The disk name can be up to 16 characters. The ID (identifier) is any two characters. You must use a unique ID for each disk you have. Don't forget that this command erases any existing data on a disk.

s:*filename* Scratches (deletes) a file from the disk.

r:*newname*=*oldname* Changes the name of file *oldname* to *newname*.

c:*backup filename=original name* Creates a new file (the backup copy) of an existing file (original copy) on the same disk.

i: Initialize disk. This resets several disk variables and should be used after you swap disks or when you have trouble reading a disk.

v: Validate disk. This recomputes the number of available blocks and can sometimes free up disk space. Always use Validate if you notice a filename on the directory flagged with an asterisk. Validate can take awhile to finish.

uj: Resets the disk drive to power-up state.

Additional Features

SpeedScript has a few commands that don't do much, but are nice to have. **CTRL-X** exchanges the character under the cursor with the character to the right of the cursor. Thus, you can fix transposition errors with a single keystroke. **CTRL-A** changes the character under the cursor from uppercase to lowercase or vice versa. You can hold down CTRL-A to continue changing the following characters.

Press **CTRL-B** to change the background and border colors. Each time you press CTRL-B, one of 16 different background colors appears. Press **CTRL-L** to cycle between one of 16 character (lettering) colors. The colors are preserved until you change them. In fact, if you exit and resave *SpeedScript*, the program will load and run with your color choice in the future.

PRINT!

If you already think *SpeedScript* has plenty of commands, wait until you see what the printing package offers. *SpeedScript* supports an array of powerful formatting features. It automatically fits your text between left and right margins that you can specify. You can center a line or block it against the right margin. *SpeedScript* skips over the perforation on continuous-form paper, or it can wait for you to insert single-sheet paper. A line of text can be printed at the top of each page (a *header)* and/or at the bottom of each page (a *footer),* and can include automatic page numbering, starting with whatever number you like.

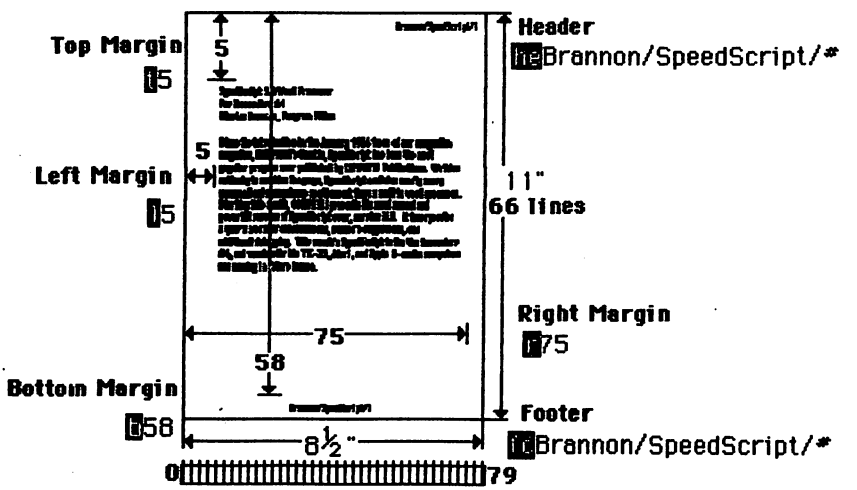
SpeedScript can print on different lengths and widths of paper, and single-, double-, triple-, or any-spacing is easy. You can print a document as big as can fit on a tape or disk by linking several files together during printing. You can print to the screen or to a sequential disk file instead of to a printer. Other features let you print to most printers using most printer interfaces, and send special codes to the printer to control features like underlining, boldfacing, and double-width type (depending on the printer).

But with all this power comes the need to learn additional commands. Fortunately, *SpeedScript* sets most of these variables to a default state. If you don't change these settings, *SpeedScript* assumes a left margin of 5, a right-margin position of 75, no header or footer, single-spacing, and continuous-paper page feeding. To begin printing, simply press **CTRL-P.** If your printer is attached, powered on, and selected (online), *SpeedScript* begins printing immediately. To cancel printing, hold down the **RUN/STOP** key until printing stops.

Figure 1-2.

Graphic Representation of Margin Settings

Values shown are default settings.



Before printing, be sure the paper in your printer is adjusted to top-of-form (move the paper perforation just above the printing element). CTRL-P assumes a Commodore printer, so it's helpful if your interface simulates the modes and codes of the Commodore 1525, MPS-801, or 1526 printers. CTRL-P prints with a device number of 4 and a secondary address of 7 (uppercase/lowercase mode).

If CTRL-P doesn't work for you, try another variation, **SHIFT-CTRL-P.** Answer the prompt "Print to: Screen, Disk, Printer?" with the single letter **S, D,** or **P.** Press any other key to cancel the command.

If you press P for printer, *SpeedScript* requests two more keystrokes. First, answer "Device number" with a number from 4 to 7. This lets you print to one of several printers addressed with different device numbers. Next, answer "Secondary Address?" with a number from 0 to 9.

Non-Commodore Printers

The secondary address is used on most non-Commodore printer interfaces to control special features. For example, you can bypass the emulation features and use graphics mode to communicate directly with your printer (see the true ASCII command below). Consult the list of secondary addresses in your printer interface manual. *SpeedScript* does not work properly with RS-232 serial printers or interfaces.

One additional note: Some printers and interfaces incorporate an automatic skip-over-perforation feature. The printer skips to the next page when it reaches the bottom of a page. Since *SpeedScript* already controls paper feeding, you need to turn off this automatic skip-over-perf feature (usually, by sending out control codes) before running *SpeedScript*, or paging won't work properly. Remember, sometimes the printer controls the skip-over-perf feature, sometimes the interface, and sometimes even both!

I've successfully tested the Commodore 64 version of *SpeedScript* with the following printers: Commodore 1525/MPS-801, Commodore 1526 (second revision), Prowriter/C. Itoh 8510, Epson MX-80, Gemini 10-X, Okimate-10, Okidata 82, Okidata 92, and Hush-80 CD.

I've also successfully tested *SpeedScript* with these printer interfaces: Cardco A/B/G+, Tymac Connection, Xetec, Turbo-Print, and MW-350.

SpeedScript should work even if your printer or interface is not on this list. These are just the ones I've tested.

Be sure your printer or interface supplies its own linefeeds. Again, consult your manuals and insure that either your printer or interface (but not both) supplies an automatic linefeed after carriage return. To test this, print a small sample of text with CTRL-P. Since the default is single-spacing, you should not see double-spacing, nor should all printing appear on the same line. If you still aren't getting linefeeds, use the linefeed command discussed below.

Printing to Screen and Disk

SHIFT–CTRL-P prints to the screen when you press S. The screen colors change to white letters on a black background, and what appears on the screen is exactly what would print on the printer. It takes two screen lines on the Commodore 64 or about four screen lines on the VIC-20 to hold one 80-column printed line, of course. If you use double-spacing (see below), it's much easier to see how each line is printed. With this screen preview, you can see where lines and pages break. To freeze printing, hold down either SHIFT key or engage SHIFT LOCK. On the 64, the border color changes to white while SHIFT is held down. When printing is finished, press any key to return to editing.

SHIFT–CTRL-P prints to a disk file when you press D. Enter the filename when requested. *SpeedScript* sends out all printer information to a sequential file. You can use other programs to process this formatted file. Try this simple example:

```
10 OPEN 1,4
20 OPEN 2,8,8,"filename"
30 GET#2,A$:SS=ST: PRINT#1,A$;: IF SS=0 THEN 30
40 PRINT#1: CLOSE1
50 CLOSE2
```

This program dumps the disk file specified by the filename in line 20 to any printer. You can use it to print *SpeedScript* files (produced with SHIFT–CTRL-P) on another Commodore computer and printer without running *SpeedScript*. Change line 10 to **OPEN 1,2,0,CHR$(6)** to dump the file to a 300-baud modem or RS-232 printer, or **OPEN 1,3** to display it on the screen.

Formatting Commands

The print formatting commands must be distinguished from normal text, so they appear onscreen in reverse field with the text and background colors switched. You enter these reverse-video letters by pressing **CTRL-£** (pound sign). (On the Commodore 64 you can also use **CTRL-3,** which is easier to type with one hand.) Answer the prompt "Enter format key:" by pressing a single key. This key is inserted into text in reverse video. All lettered printer commands should be entered in lowercase (unSHIFTed). During printing, *SpeedScript* treats these characters as printing commands. (See Appendix D for a quick-reference chart of format commands.)

There are two kinds of printing commands, which I'll call Stage 1 and Stage 2. Stage 1 commands usually control variables such as left margin and right margin. Most are followed by a number, with no space between the command and the number. Stage 1 commands are executed before a line is printed.

Stage 2 commands, like centering and underlining, are executed while the line is being printed. Usually, Stage 1 commands must be on a line of their own, although you can group several Stage 1 commands together on a line. Stage 2 commands are by nature embedded within a line of text. A sample Stage 1 line could look like this:

```
l10r50s2
```

Embedded Stage 2 commands look like this:

```
cThis line is centered.←
This is uunderliningu.←
```

Stage 1 Commands

l Left margin. Follow with a number from 0 to 255. Use 0 for no margin. Defaults to 5. See Figure 1-2 for a graphic illustration of margin settings.

r Right margin position, a number from 1 to 255. Defaults to 75. Be sure the right-margin value is greater than the left-margin value, or *SpeedScript* will go bonkers. Some printer interfaces force a certain printing width, usually 80 characters

wide. You'll need to disable this in order to permit *SpeedScript* to print lines longer than 80 characters.

t Top margin. The position at which the first line of text is printed, relative to the top of the page. Defaults to 5. The header (if any) is always printed on the first line of the page, before the first line of text.

b Bottom margin. The line at which printing stops before continuing to the next page. Standard 8-1/2 × 11 inch paper has 66 lines. Bottom margin defaults to the fifty-eighth line. The footer (if any) is always printed on the last line of the page, after the last line of text.

p Page length. Defaults to 66. If your printer does not print 6 lines per inch, multiply lines-per-inch by 11 to get the page length. European paper is usually longer than American paper—11-5/8 or 12 inches. Try a page length of 69 or 72.

s Spacing. Defaults to single-spacing. Follow with a number from 1 to 255. Use 1 for single-spacing, 2 for double-spacing, 3 for triple-spacing.

@ Start numbering *at* page number given. Page numbering normally starts with 1.

? Disables printing until selected page number is reached. For example, a value of 3 would start printing the third page of your document. Normally, *SpeedScript* starts printing with the first page.

x Sets the page width, in columns (think *a cross).* Defaults to 80. You need to change this for the sake of the centering command if you are printing in double-width or condensed type, or if you're using a 40-column or wide-carriage printer.

n Forced paging. Normally, *SpeedScript* prints the footer and moves on to the next page only when it has finished a page, but you can force it to continue to the next page by issuing this command. It requires no numbers.

m Margin release. Disables the left margin for the next printed line. Remember that this executes before the line is printed. It's used for outdenting.

a True ASCII. Every character is assigned a number in the ASCII (American Standard Code for Information Interchange) character set. Most printers use this true ASCII standard, but Commodore printers exchange the values for

uppercase and lowercase to match Commodore's own variation of ASCII. Some printer interfaces do not translate Commodore ASCII into true ASCII, so you need to use this command to tell *SpeedScript* to translate. Also, you will sometimes want to intentionally disable your interface's emulation mode in order to control special printer features that would otherwise be rejected by emulation. Place this command as the first character in your document, even before the header and footer definitions. Don't follow it with a number.

Since, in effect, the true ASCII command changes the case of all letters, you can type something in lowercase and use true ASCII to make it come out in uppercase.

w Page wait. Like the true ASCII command, this one should be placed at the beginning of your document before any text. With page wait turned on, *SpeedScript* prompts you to "Insert next sheet, press RETURN" when each page is finished printing. Insert the next sheet, line it up with the printhead, then press RETURN to continue. Page wait is ignored during disk or screen output.

j Select automatic linefeeds after carriage return. Like **a** and **w**, this command must be placed before any text. Don't use this command to achieve double-spacing, but only if all text prints on the same line.

i Information. This works like REM in BASIC. You follow the command with a line of text, up to 255 characters, ending in a return mark. This line will be ignored during printing; it's handy for making notes to yourself such as the filename of the document.

h Header define and enable. The header must be a single line of text ending with a return mark (up to 254 characters). The header prints on the first line of each page. You can include Stage 2 commands such as centering and page numbering in a header. You can use a header by itself without a footer. The header and footer should be defined at the top of your document, before any text. If you want to prevent the header from printing on the first page, put a return mark by itself at the top of your document before the header definition.

f Footer define and enable. The footer must be a single line of text ending in a return mark (up to 254 characters). The

footer prints on the last line of each page. As with the header, you can include Stage 2 printing commands, and you don't need to set the header to use a footer.

g GOTO (link) next file. Put this command as the last line in your document. Follow the command with the letter *D* for disk or *T* for tape, then a colon (:), then the name of the file to print next. After the text in memory is printed, the link command loads the next file into memory. You can continue linking in successive files, but don't include a link in the last file. Before you start printing a linked file, make sure the first of the linked files is in memory. When printing is finished, the last file linked to will be in memory.

Stage 2 Commands

These commands either precede a line of text or are embedded within one.

c Centering. Put this at the beginning of a line you want to center. This will center only one line, ending in a return mark. Repeat this command at the beginning of every line you want centered. Centering uses the page-width setting (see above) to properly center the line. To center a double-width line, either set the page width to 40 or pad out the rest of the line with an equal number of spaces. If you use double-width, remember that the spaces preceding the centered text will be double-wide spaces.

When *SpeedScript* encounters this command, it prints the current page number. You usually embed this within a header or footer.

u A simple form of underlining. It does not work on Commodore printers, but only on printers that recognize CHR$(8) as a backspace and CHR$(95) as an underline character. Underlining works on spaces, too. Use the first **u** to start underlining and another one to turn off underlining.

Fonts and Styles

Most dot-matrix printers are capable of more than just printing text at ten characters per inch. The Commodore MPS-801 can print in double-width and reverse field. Some printers have several character sets, with italics and foreign language characters. Most can print in double-width (40 characters per line),

condensed (132 characters per line), and in either pica or elite. Other features include programmable characters, programmable tab stops, and graphics modes. Many word processors customize themselves to a particular printer, but *SpeedScript* was purposely designed not to be printer-specific. Instead, *SpeedScript* lets you define your own Stage 2 printing commands.

You define a programmable *printkey* by choosing any character that is not already used for other printer commands. The entire uppercase alphabet is available for printkeys, and you can choose letters that are related to their function (like D for double-width). You enter these commands like printer commands, by first pressing **CTRL-3** on the Commodore 64 or **CTRL-£** on the VIC-20.

To define a printkey, just press CTRL-3 (64) or CTRL-£ (VIC), then the key you want to assign as the printkey, then an equal sign (=), and finally the ASCII value to be substituted for the printkey during printing. For example, to define the + key as the letter Z, you first look up the ASCII value of the letter Z (in either your printer manual or user's manual). The ASCII value of the letter Z is 91, so the definition is

```
■=91←
```

Now, anywhere you want to print the letter Z, substitute the printkey:

```
Gad■ooks! The ■oo is ■any!←
```

This would appear on paper as

```
Gadzooks! The zoo is zany!
```

More practically, look up the value of reverse-on and reverse-off. Reverse-on, a value of 18, prints all text in reverse video until canceled by reverse-off (a value of 146) or a carriage return. So define SHIFT-R as 18 and SHIFT-O as 146. Anywhere you want to print a word in reverse, bracket the word with printkey R and printkey O.

You can similarly define whatever codes your printer uses for features like double-width or emphasized mode. For your convenience, four of the printkeys are predefined, though you

can change them. Printkey 1 is defined as a 27, the value of the ESCape code used to precede many two-character printer commands. (With some printer interfaces, you must send two ESCape codes to bypass the interface's emulation.) For example, the Epson command for double strike is ESC-G. You can select it in *SpeedScript* with

```
1G
```

Printkey 2, a value of 14, goes into double-width mode on most printers, and printkey 3, a value of 15, turns off double-width on some printers and selects condensed mode on others. Printkey 4 is defined as 18, which selects reverse field with Commodore printers (and on some graphics interfaces in emulation mode) or condensed mode on some other printers.

With so many codes available, you can even design custom logos and symbols using your printer's graphics mode. For example, on the 1525/MPS-801, you can draw a box (perhaps for a checklist) by first setting the appropriate codes:

```
1=82=253=2554=193←
```

Then display the box with text by typing

```
13444432 Toothpaste←
```

This appears on paper as

```
□ Toothpaste
```

Keep one thing in mind about printkeys. *SpeedScript* always assumes it is printing to a rather dumb, featureless printer, the least common denominator. *SpeedScript* doesn't understand the intent of a printkey; it justs sends its value out. So if you make one word within a line double-width, it may make the line overflow the specified right margin. There's no way for *SpeedScript* to include built-in font and type-style codes without being customized for a particular printer since no set of codes is universal to all printers.

Hints and Tips

It may take you awhile to fully master *SpeedScript*, but as you do you'll discover many ways to use the editing and formatting commands. For example, there is a simple way to simulate tab stops, say, for a columnar table. Just type a period at every tab-stop position. Erase the line, then restore it multiple times. When you are filling in the table, just use word left/word right to jump quickly between the periods. Or you can use the programmable printkeys to embed your printer's own commands for setting and jumping to tab stops.

You don't have to change or define printer commands every time you write. Just save these definitions as a small text file, and load this file in each time you write. You can create many custom definition files and have them ready to use on disk. You can create customized "fill-in-the-blank" letters. Just type the letter, and everywhere you'll need to insert something, substitute a graphics symbol. When you're ready to customize the letter, just hunt for each graphics symbol and insert the specific information.

SpeedScript does not work with any 80-column video boards or software 80-column emulators. *SpeedScript* also wipes out most kinds of resident (RAM-loaded) software, including most software-simulated printer drivers. However, you can print to disk using SHIFT–CTRL-P, then dump the disk file to the printer from BASIC.

Chapter 2
Entering *SpeedScript*

Typing In BASIC Programs

In order to make the typing in of *SpeedScript* as easy as possible, some program entry aids written in BASIC have been included. In addition, there are a number of other programs in this book which are also written in BASIC. In order to assist you in understanding how to enter these programs, COMPUTE! has established the following listing conventions.

Generally, VIC or 64 program listings will contain words within braces which spell out any special characters: {DOWN} would mean to press the cursor-down key; {5 SPACES} would mean to press the space bar five times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listings. For example, S would mean to type the S key while holding the SHIFT key. This would appear on your screen as a heart symbol. If you find an underlined key enclosed in braces (for example, {10 N}), you should type the key as many times as indicated. In that case, you would enter ten shifted N's.

If a key is enclosed in special brackets, [< >], you should hold down the *Commodore key* while pressing the key inside the special brackets. (The Commodore key is the key in the lower-left corner of the keyboard.) Again, if the key is preceded by a number, you should press the key as many times as necessary.

Rarely, in programs for the 64, you'll see a solitary letter of the alphabet enclosed in braces. These characters can be entered by holding down the CTRL key while typing the letter in the braces. For example, {A} would indicate that you should press CTRL-A. You should never have to enter such a character on the VIC.

Quote Mode

You know that you can move the cursor around the screen with the CRSR keys. Sometimes a programmer will want to move the cursor under program control. That's why you see all the {LEFT}'s, {HOME}'s, and {BLU}'s in our programs. The only way the computer can tell the difference between direct and programmed cursor control is the quote mode.

Once you press the quote (the double quote, SHIFT-2), you are in the quote mode. If you type something and then try to change it by moving the cursor left, you'll only get a bunch of reverse-video lines. These are the symbols for cursor left. The only editing key that isn't programmable is the INST/DEL key; you can still use INST/DEL to back up and edit the line. Once you type another quote, you are out of quote mode.

You also go into quote mode when you INSerT spaces into a line. In any case, the easiest way to get out of quote mode is just to press RETURN. You'll then be out of quote mode and you can cursor up to the mistyped line and fix it.

Refer to the following table when entering cursor and color control keys:

When You Read:	Press:	See:
{CLR}	SHIFT CLR/HOME	
{HOME}	CLR/HOME	
{UP}	SHIFT ↑ CRSR ↓	
{DOWN}	↑ CRSR ↓	
{LEFT}	SHIFT ← CRSR →	
{RIGHT}	← CRSR →	
{RVS}	CTRL 9	
{OFF}	CTRL 0	
{BLK}	CTRL 1	
{WHT}	CTRL 2	
{RED}	CTRL 3	
{CYN}	CTRL 4	
{PUR}	CTRL 5	
{GRN}	CTRL 6	
{BLU}	CTRL 7	
{YEL}	CTRL 8	

When You Read:	Press:	See:
{ 1 }	COMMODORE 1	
{ 2 }	COMMODORE 2	
{ 3 }	COMMODORE 3	
{ 4 }	COMMODORE 4	
{ 5 }	COMMODORE 5	
{ 6 }	COMMODORE 6	
{ 7 }	COMMODORE 7	
{ 8 }	COMMODORE 8	
{ F1 }	f1	
{ F2 }	SHIFT f1	
{ F3 }	f3	
{ F4 }	SHIFT f3	
{ F5 }	f5	
{ F6 }	SHIFT f5	
{ F7 }	f7	
{ F8 }	SHIFT f7	
←	←	
↑	SHIFT ↑	

The Machine Language Editor: MLX

Remember the last time you typed in the BASIC loader for a long machine language program? You typed in hundreds of numbers and commas. Even then, you couldn't be sure if you typed it in right. So you went back, proofread, tried to run the program, crashed, went back again, proofread, corrected a few typing errors, ran again, crashed again, rechecked your typing. Frustrating, wasn't it?

Now, "MLX" comes to the rescue. MLX makes it easy to enter *SpeedScript* and all those long machine language programs with a minimum of fuss. It lets you enter the numbers from a special list that looks similar to DATA statements, and it checks your typing on a line-by-line basis. It won't let you enter illegal characters when you should be typing numbers. It won't let you enter numbers greater than 255. It will prevent you from entering the numbers on the wrong line. In short, MLX will make proofreading obsolete.

Tape or Disk Copies

In addition, MLX will generate a ready-to-use tape or disk copy of *SpeedScript*. You can then use the LOAD command to read the program into the computer, just like you would with a BASIC program. Specifically, you enter LOAD "SPEEDSCRIPT",1 (for tape) or LOAD "SPEEDSCRIPT",8 (for disk). To start *SpeedScript* once it is loaded, simply type RUN and press RETURN. (If you use MLX to save other machine language programs, you'll usually need to add ,1 to the above LOAD command and enter a SYS command to transfer control from BASIC to your machine language program.)

Using MLX

Type in and save the MLX version for your computer; if you are using a VIC-20 you must have 8K or more expansion memory attached. (As mentioned above, MLX can be used for other machine language programs from other COMPUTE! publications.)

Before you begin typing *SpeedScript* (or begin a subsequent session of typing if you enter *SpeedScript* in more than

one sitting), you must enter certain POKEs *before* you load and run the MLX program (be sure you use the proper POKEs for your computer). These POKEs are essential to protect *SpeedScript* from BASIC while you are typing it in. Again, these POKEs should be performed *before* you load MLX, but they are not necessary to run the finished program:

Commodore 64:

POKE 44,33:POKE 8448,0:NEW

VIC-20:

POKE 44,42:POKE 10752,0:NEW

Once the proper POKEs have been entered, you are ready to load and run the MLX program. Once running, MLX will ask you for two numbers: the starting address and the ending address. Then you'll get a prompt showing the specified starting address; that tells you to type in the corresponding first line of the program.

If you are using a Commodore 64, enter:

Starting Address? 2049
Ending Address? 8204

If you are using a VIC-20, enter:

Starting Address? 4609
Ending Address? 10482

You will then see the first prompt, the number 2049, on the Commodore 64, or the number 4609, on the VIC-20, followed by a colon. Type in each three-digit number shown in the listing. You do not need to press the comma shown in the listing. MLX types the comma automatically.

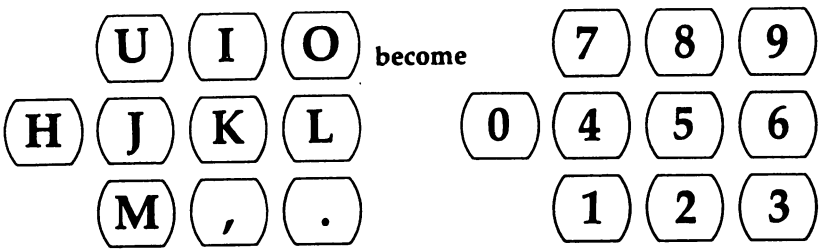
Subsequent prompts will ask you to type in additional lines from the MLX listing. Each line is six numbers plus a checksum. If you enter any of the six numbers wrong or enter the checksum wrong, the computer will sound a buzzer and prompt you to reenter the entire line. If you enter the line correctly, a pleasant bell tone will sound and you may go on to enter the next line.

A Special Editor

You are not using the normal BASIC editor with MLX. For example, it will only accept numbers as input. If you make a typing error, press the INST/DEL key; the entire number will

be deleted. You can press it as many times as necessary, back to the start of the line. If you enter three-digit numbers as listed, the computer automatically prints the comma and goes on to accept the next number. If you enter less than three digits, you can press either the space bar or RETURN key to advance to the next number. The checksum automatically appears in reverse video for emphasis.

To make it even easier to enter these numbers, MLX redefines part of the keyboard as a numeric keypad (lines 581–584).



When testing it, I've found MLX to be an extremely easy way to enter long listings. With the audio cues provided, you don't even have to look at the screen if you're a touch-typist.

Done at Last!

When you get through typing, assuming you typed *SpeedScript* in one session, you can then save the completed and bug-free program to tape or disk. Follow the instructions displayed on the screen. If you get any error messages while saving, you probably have a bad disk, a full disk, or a typo in MLX (a rare occurrence if you use the "The Automatic Proofreader"—see the next article).

Command Control

What if you don't want to enter the whole program in one sitting? MLX lets you enter as much as you want, save the completed portion, and then reload your work from tape or disk when you want to continue. MLX recognizes these commands:

SHIFT-S: Save
SHIFT-L: Load
SHIFT-N: New Address
SHIFT-D: Display

Hold down SHIFT while you press the appropriate key. You will jump out of the line you've been typing, so I recommend you do it at a prompt. Use the Save command to store what you've been working on. It will write the tape or disk file as if you've finished. Remember what address you stop on. Then, the next time you run MLX, answer all the prompts as you did before and insert the disk or tape containing the stored file. When you get the entry prompt, press SHIFT-L to reload the file into memory. You'll then use the New Address command (SHIFT-N) to resume typing.

New Address and Display

After you press SHIFT-N, enter the address where you previously stopped. The prompt will change and you can continue typing. Always enter a New Address that matches up with one of the line numbers in the special listing, or the checksums won't match up. You can use the Display command to display a section of your typing. After you press SHIFT-D, enter two addresses within the line number range of the listing. You can stop the display by pressing any key.

I hope you will find MLX to be a true labor-saving program. Since it has been tested by entering actual programs, you can count on it as an aid for generating bug-free machine language. Be sure to save MLX; it will be used for future applications in other COMPUTE! books.

Program 2-1. Commodore 64 MLX

For mistake-proof program entry, be sure to read "The Automatic Proofreader," later in this chapter.

```
10 REM LINES CHANGED FROM MLX VERSION 2.00 ARE 750
   ,765,770 AND 860                              :rem 50
20 REM LINE CHANGED FROM MLX VERSION 2.01 IS 300
                                                 :rem 147
30 REM LINE CHANGED FROM MLX VERSION 2.02 IS 763
                                                 :rem 162
100 PRINT"{CLR}{6}";CHR$(142);CHR$(8);:POKE53281,1
    :POKE53280,1                                  :rem 67
101 POKE 788,52:REM DISABLE RUN/STOP             :rem 119
200 PRINT"{2 DOWN}{PUR}{BLK} MACHINE LANGUAGE EDIT
    OR VERSION 2.03{5 DOWN}"                     :rem 239
210 PRINT"{5}{2 UP}STARTING ADDRESS?{8 SPACES}
    {9 LEFT}";                                   :rem 143
```

```
215 INPUTS:F=1-F:C$=CHR$(31+119*F)          :rem 166
220 IFS<256OR(S>40960ANDS<49152)ORS>53247THENGOSUB
    3000:GOTO210                            :rem 235
225 PRINT:PRINT:PRINT                       :rem 180
230 PRINT"{5}{2 UP}ENDING ADDRESS?{8 SPACES}
    {9 LEFT}";:INPUTE:F=1-F:C$=CHR$(31+119*F)
                                            :rem 20
240 IFE<256OR(E>40960ANDE<49152)ORE>53247THENGOSUB
    3000:GOTO230                            :rem 183
250 IFE<STHENPRINTC$;"{RVS}ENDING < START
    {2 SPACES}":GOSUB1000:GOTO 230          :rem 176
260 PRINT:PRINT:PRINT                       :rem 179
300 PRINT"{CLR}";CHR$(14):AD=S               :rem 56
310 A=1:PRINTRIGHT$("0000"+MID$(STR$(AD),2),5);":"
    ;                                       :rem 33
315 FORJ=ATO6                               :rem 33
320 GOSUB570:IFN=-1THENJ=J+N:GOTO320        :rem 228
390 IFN=-211THEN 710                        :rem 62
400 IFN=-204THEN 790                        :rem 64
410 IFN=-206THENPRINT:INPUT"{DOWN}ENTER NEW ADDRES
    S";ZZ                                   :rem 44
415 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"{RVS}OUT OF
    {SPACE}RANGE":GOSUB1000:GOTO410         :rem 225
417 IFN=-206THENAD=ZZ:PRINT:GOTO310         :rem 238
420 IF N<>-196 THEN 480                     :rem 133
430 PRINT:INPUT"DISPLAY:FROM";F:PRINT,"TO";:INPUTT
                                            :rem 234
440 IFF<SORF>EORT<SORT>ETHENPRINT"AT LEAST";S;"
    {LEFT}, NOT MORE THAN";E:GOTO430        :rem 159
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$("0000"+MID$(S
    TR$(I),2),5);":";                       :rem 30
451 FORK=0TO5:N=PEEK(I+K):PRINTRIGHT$("00"+MID$(ST
    R$(N),2),3);",";                        :rem 66
460 GETA$:IFA$=""THENPRINT:PRINT:GOTO310    :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRINT:GOTO310
                                            :rem 50
480 IFN<0 THEN PRINT:GOTO310                :rem 168
490 A(J)=N:NEXTJ                            :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CKSUM=(CKSU
    M+A(I))AND255:NEXT                      :rem 200
510 PRINTCHR$(18);·GOSUB570:PRINTCHR$(146);:rem 94
511 IFN=-1THENA=6:GOTO315                   :rem 254
515 PRINTCHR$(20):IFN=CKSUMTHEN530          :rem 122
520 PRINT:PRINT"LINE ENTERED WRONG : RE-ENTER":PRI
    NT:GOSUB1000:GOTO310                    :rem 176
530 GOSUB2000                               :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT:POKE54272,0:POK
    E54273,0                                :rem 227
```

```
550 AD=AD+6:IF AD<E THEN 310                          :rem 212
560 GOTO 710                                          :rem 108
570 N=0:Z=0                                            :rem 88
580 PRINT"{£}";                                        :rem 81
581 GETA$:IFA$=""THEN581                               :rem 95
582 AV=-(A$="M")-2*(A$=",")-3*(A$=".")-4*(A$="J")-
    5*(A$="K")-6*(A$="L")                              :rem 41
583 AV=AV-7*(A$="U")-8*(A$="I")-9*(A$="O"):IFA$="H
    "THENA$="0"                                        :rem 134
584 IFAV>0THENA$=CHR$(48+AV)                           :rem 134
585 PRINTCHR$(20);:A=ASC(A$):IFA=13ORA=44ORA=32THE
    N670                                               :rem 229
590 IFA>128THENN=-A:RETURN                             :rem 137
600 IFA<>20 THEN 630                                    :rem 10
610 GOSUB690:IFI=1ANDT=44THENN=-1:PRINT"{OFF}
    {LEFT} {LEFT}";:GOTO690                            :rem 62
620 GOTO570                                            :rem 109
630 IFA<48ORA>57THEN580                                :rem 105
640 PRINTA$;:N=N*10+A-48                               :rem 106
650 IFN>255 THEN A=20:GOSUB1000:GOTO600                :rem 229
660 Z=Z+1:IFZ<3THEN580                                  :rem 71
670 IFZ=0THENGOSUB1000:GOTO570                         :rem 114
680 PRINT",";:RETURN                                   :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)               :rem 149
691 FORI=1TO3:T=PEEK(S%-I)                              :rem 67
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT                :rem 205
700 PRINTLEFT$("{3 LEFT}",I-1);:RETURN                   :rem 7
710 PRINT"{CLR}{RVS}*** SAVE ***{3 DOWN}"              :rem 236
715 PRINT"{2 DOWN}(PRESS {RVS}RETURN{OFF} ALONE TO
     CANCEL SAVE){DOWN}"                               :rem 106
720 F$="":INPUT"{DOWN} FILENAME";F$:IFF$=""THENPRI
    NT:PRINT:GOTO310                                   :rem 71
730 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR {RVS}D
    {OFF}ISK: (T/D)"                                   :rem 228
740 GETA$:IFA$<>"T"ANDA$<>"D"THEN740                    :rem 36
750 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$:OPEN15,8,
    15,"S"+F$:CLOSE15                                  :rem 212
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$):POKE782
    ,ZK/256                                             :rem 3
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(T$):SYS65
    469                                                :rem 109
763 POKE780,1:POKE781,DV:POKE782,0:SYS65466:rem 68
765 K=S:POKE254,K/256:POKE253,K-PEEK(254)*256:POKE
    780,253                                            :rem 17
766 K=E+1:POKE782,K/256:POKE781,K-PEEK(782)*256:SY
    S65496                                             :rem 235
770 IF(PEEK(783)AND1)OR(191ANDST)THEN780               :rem 111
775 PRINT"{DOWN}DONE.{DOWN}":GOTO310                   :rem 113
780 PRINT"{DOWN}ERROR ON SAVE.{2 SPACES}TRY AGAIN.
    ":IFDV=1THEN720                                    :rem 171
```

```
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$;E2$:CLOS
    E15:GOTO720                                    :rem 103
790 PRINT"{CLR}{RVS}*** LOAD ***{2 DOWN}"          :rem 212
795 PRINT"{2 DOWN}(PRESS {RVS}RETURN{OFF} ALONE TO
     CANCEL LOAD)"                                 :rem 82
800 F$="":INPUT"{2 DOWN} FILENAME";F$:IFF$=""THENP
    RINT:GOTO310                                   :rem 144
810 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR {RVS}D
    {OFF}ISK: (T/D)"                               :rem 227
820 GETA$:IFA$<>"T"ANDA$<>"D"THEN820               :rem 34
830 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$           :rem 157
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$):POKE782
    ,ZK/256                                        :rem 2
841 POKE781,ZK-PEEK(782)*256:POKE780,LEN(T$):SYS65
    469                                            :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS65466:rem 70
850 POKE780,0:SYS65493                             :rem 11
860 IF(PEEK(783)AND1)OR(191ANDST)THEN870           :rem 111
865 PRINT"{DOWN}DONE.":GOTO310                     :rem 96
870 PRINT"{DOWN}ERROR ON LOAD.{2 SPACES}TRY AGAIN.
    {DOWN}":IFDV=1THEN800                          :rem 172
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$;E2$:CLOS
    E15:GOTO800                                    :rem 102
1000 REM BUZZER                                    :rem 135
1001 POKE54296,15:POKE54277,45:POKE54278,165
                                                   :rem 207
1002 POKE54276,33:POKE 54273,6:POKE54272,5         :rem 42
1003 FORT=1TO200:NEXT:POKE54276,32:POKE54273,0:POK
     E54272,0:RETURN                               :rem 202
2000 REM BELL SOUND                                 :rem 78
2001 POKE54296,15:POKE54277,0:POKE54278,247
                                                   :rem 152
2002 POKE 54276,17:POKE54273,40:POKE54272,0:rem 86
2003 FORT=1TO100:NEXT:POKE54276,16:RETURN          :rem 57
3000 PRINTC$;"{RVS}NOT ZERO PAGE OR ROM":GOTO1000
                                                   :rem 89
```

Program 2-2. VIC-20 MLX

For mistake-proof program entry, be sure to read "The Automatic Proofreader," later in this chapter.

```
10 REM LINES CHANGED FROM VIC MLX VERSION 2.00 ARE
    581,582,765                                    :rem 166
100 PRINT"{CLR}{PUR}";CHR$(142);CHR$(8);           :rem 181
101 POKE 788,194:REM DISABLE RUN/STOP              :rem 174
110 PRINT"{RVS}{14 SPACES}"                        :rem 117
120 PRINT"{RVS} {RIGHT}{OFF}[*]£{RVS}{RIGHT}
    {RIGHT}{2 SPACES}[*]{OFF}[*]£{RVS}£{RVS} "
                                                   :rem 191
```

```
130 PRINT"{RVS} {RIGHT} [G]{RIGHT} {2 RIGHT} {OFF}
    £{RVS}£[*]{OFF}[*]{RVS} "                    :rem 232
140 PRINT"{RVS}{14 SPACES}"                      :rem 120
200 PRINT"{2 DOWN}{PUR}{BLK}MACHINE LANGUAGE":PRIN
    T"EDITOR VER 2.02{5 DOWN}"                   :rem 192
210 PRINT"{BLK}{3 UP}STARTING ADDRESS":INPUTS:F=1-
    F:C$=CHR$(31+119*F)                          :rem 97
220 IFS<256ORS>32767THENGOSUB3000:GOTO210        :rem 2
225 PRINT:PRINT:PRINT:PRINT                      :rem 123
230 PRINT"{BLK}{3 UP}ENDING ADDRESS":INPUTE:F=1-F:
    C$=CHR$(31+119*F)                            :rem 158
240 IFE<256ORE>32767THENGOSUB3000:GOTO230        :rem 234
250 IFE<STHENPRINTC$;"{RVS}ENDING < START
    {2 SPACES}":GOSUB1000:GOTO 230               :rem 176
260 PRINT:PRINT:PRINT                            :rem 179
300 PRINT"{CLR}";CHR$(14):AD=S                   :rem 56
310 A=1:PRINTRIGHT$("0000"+MID$(STR$(AD),2),5);":"
    ;                                            :rem 33
315 FOR J=A TO 6                                 :rem 33
320 GOSUB570:IFN=-1THENJ=J+N:GOTO320             :rem 228
390 IFN=-211THEN 710                             :rem 62
400 IFN=-204THEN 790                             :rem 64
410 IFN=-206THENPRINT:INPUT"{DOWN}ENTER NEW ADDRES
    S";ZZ                                        :rem 44
415 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"{RVS}OUT OF
    {SPACE}RANGE":GOSUB1000:GOTO410              :rem 225
417 IFN=-206THENAD=ZZ:PRINT:GOTO310              :rem 238
420 IF N<>-196 THEN 480                          :rem 133
430 PRINT:INPUT"DISPLAY:FROM";F:PRINT,"TO";:INPUTT
                                                 :rem 234
440 IFF<SORF>EORT<SORT>ETHENPRINT"AT LEAST";S;"
    {LEFT}, NOT MORE THAN";E:GOTO430             :rem 159
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$("0000"+MID$(S
    TR$(I),2),5);":";                            :rem 30
455 FORK=0TO5:N=PEEK(I+K):IFK=3THENPRINTSPC(10);
                                                 :rem 34
457 PRINTRIGHT$("00"+MID$(STR$(N),2),3);",";
                                                 :rem 157
460 GETA$:IFA$>""THENPRINT:PRINT:GOTO310         :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRINT:GOTO310
                                                 :rem 50
480 IFN<0 THEN PRINT:GOTO310                     :rem 168
490 A(J)=N:NEXTJ                                 :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CKSUM=(CKSU
    M+A(I))AND255:NEXT                           :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR$(146);:rem 94
511 IFN=-1THENA=6:GOTO315                        :rem 254
515 PRINTCHR$(20):IFN=CKSUMTHEN530               :rem 122
520 PRINT:PRINT"LINE ENTERED WRONG":PRINT"RE-ENTER
    ":PRINT:GOSUB1000:GOTO310                    :rem 129
```

```
530 GOSUB2000                                          :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT                      :rem 80
550 AD=AD+6:IF AD<E THEN 310                           :rem 212
560 GOTO 710                                           :rem 108
570 N=0:Z=0                                             :rem 88
580 PRINT"{+}";                                         :rem 79
581 GETA$:IFA$=""THEN581                                :rem 95
582 AV=-(A$="M")-2*(A$=",")-3*(A$=".")-4*(A$="J")-
    5*(A$="K")-6*(A$="L")                               :rem 41
583 AV=AV-7*(A$="U")-8*(A$="I")-9*(A$="O"):IFA$="H
    "THENA$="0"                                        :rem 134
584 IFAV>0THENA$=CHR$(48+AV)                           :rem 134
585 PRINTCHR$(20);:A=ASC(A$):IFA=13ORA=44ORA=32THE
    N670                                               :rem 229
590 IFA>128THENN=-A:RETURN                             :rem 137
600 IFA<>20 THEN 630                                    :rem 10
610 PRINTCHR$(146);:GOSUB690:IFI=1ANDT=44THENN=-1:
    PRINT"{LEFT} {LEFT}";:GOTO690                      :rem 155
620 GOTO570                                            :rem 109
630 IFA<48ORA>57THEN580                                :rem 105
640 PRINTA$;:N=N*10+A-48                               :rem 106
650 IFN>255 THEN A=20:GOSUB1000:GOTO600                :rem 229
660 Z=Z+1:IFZ<3THEN580                                  :rem 71
670 IFZ=0THENGOSUB1000:GOTO570                         :rem 114
680 PRINT",";:RETURN                                   :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)               :rem 149
692 FORI=1TO3:T=PEEK(S%-I)                              :rem 68
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT                :rem 205
700 PRINTLEFT$("{3 LEFT}",I-1);:RETURN                   :rem 7
710 PRINT"{CLR}{RVS}*** SAVE ***{3 DOWN}"              :rem 236
720 F$="":INPUT"{DOWN} FILENAME";F$:IFF$=""THEN310
                                                       :rem 128
730 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR {RVS}D
    {OFF}ISK: (T/D)"                                   :rem 228
740 GETA$:IFA$<>"T"ANDA$<>"D"THEN740                    :rem 36
750 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$:OPEN15,8,
    15,"S"+F$:CLOSE15                                  :rem 212
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$):POKE782
    ,ZK/256                                             :rem 3
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(T$):SYS65
    469                                                :rem 109
763 POKE780,1:POKE781,DV:POKE782,1:SYS65466:rem 69
765 K=S:POKE254,K/256:POKE253,K-PEEK(254)*256:POKE
    780,253                                             :rem 17
766 K=E+1:POKE782,K/256:POKE781,K-PEEK(782)*256:SY
    S65496                                             :rem 235
770 IF(PEEK(783)AND1)OR(191ANDST)THEN780               :rem 111
775 PRINT"{DOWN}DONE.":GOTO310                          :rem 96
780 PRINT"{DOWN}ERROR ON SAVE.{2 SPACES}TRY AGAIN.
    ":IFDV=1THEN720                                    :rem 171
```

```
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$;E2$:CLOS
    E15:GOTO720                                    :rem 103
782 GOTO720                                        :rem 115
790 PRINT"{CLR}{RVS}*** LOAD ***{2 DOWN}"          :rem 212
800 F$="":INPUT"{2 DOWN} FILENAME";F$:IFF$=""THEN3
    10                                             :rem 144
810 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR {RVS}D
    {OFF}ISK: (T/D)"                               :rem 227
820 GETA$:IFA$<>"T"ANDA$<>"D"THEN820               :rem 34
830 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$           :rem 157
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$):POKE782
    ,ZK/256                                        :rem 2
841 POKE781,ZK-PEEK(782)*256:POKE780,LEN(T$):SYS65
    469                                            :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS65466:rem 70
850 POKE780,0:SYS65493                             :rem 11
860 IF(PEEK(783)AND1)OR(191ANDST)THEN870           :rem 111
865 PRINT"{DOWN}DONE.":GOTO310                     :rem 96
870 PRINT"{DOWN}ERROR ON LOAD.{2 SPACES}TRY AGAIN.
    {DOWN}":IFDV=1THEN800                          :rem 172
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$;E2$:CLOS
    E15:GOTO800                                    :rem 102
1000 REM BUZZER                                    :rem 135
1001 POKE36878,15:POKE36874,190                    :rem 206
1002 FORW=1TO300:NEXTW                             :rem 117
1003 POKE36878,0:POKE36874,0:RETURN                :rem 74
2000 REM BELL SOUND                                :rem 78
2001 FORW=15TO0STEP-1:POKE36878,W:POKE36876,240:NE
     XTW                                           :rem 22
2002 POKE36876,0:RETURN                            :rem 119
3000 PRINTC$;"{RVS}NOT ZERO PAGE OR ROM":GOTO1000
                                                   :rem 89
```

The Automatic Proofreader

"The Automatic Proofreader" will help you type in BASIC program listings without typing mistakes. It is a short error-checking program that hides itself in memory. When activated, it lets you know immediately after you type a line from a program listing if you have made a mistake. Please read these instructions carefully before typing any programs in this book.

Preparing the Proofreader

1. Using Program 2-3, below, type in the Proofreader. Be very careful when entering the DATA statements—don't type an *l* instead of a *1*, an *O* instead of a *0*, extra commas, and so forth.
2. Save the Proofreader on tape or disk at least twice *before running it for the first time.* This is very important because the Proofreader erases part of itself when you first type RUN.
3. After the Proofreader is saved, type RUN. It will check itself for typing errors in the DATA statements and warn you if there's a mistake. Correct any errors and save the corrected version. Keep a copy in a safe place—you'll need it again and again, every time you enter a program from a COMPUTE! publication.
4. When a correct version of the Proofreader is run, it activates itself. You are now ready to enter a program listing. If you press RUN/STOP-RESTORE, the Proofreader is disabled. To reactivate it, just type the command SYS 886 and press RETURN.

Using the Proofreader

The MLX listings in this book have a *checksum number* appended to the end of each line, for example, *:rem 123. Don't enter this statement when typing in a program.* It is just for your information. The rem makes the number harmless if someone does type it in. It will, however, use up memory if you enter it, and it will confuse the Proofreader, even if you entered the rest of the line correctly.

When you type in a line from a program listing and press RETURN, the Proofreader displays a number at the top of your screen. *This checksum number must match the checksum*

number in the printed listing. If it doesn't, it means you typed the line differently from the way it is listed. Immediately recheck your typing. Remember, don't type the rem statement with the checksum number; it is published only so you can check it against the number which appears on your screen.

The Proofreader is not picky about spaces. It will not notice extra spaces or missing ones. This is for your convenience, since spacing is generally not important. But since occasionally proper spacing *is* important, be extra careful with spaces.

Due to the nature of a checksum, the Proofreader will not catch all errors. Since 1 + 3 + 5 = 3 + 1 + 5, the Proofreader cannot catch errors of transposition. Thus, the Proofreader will not notice if you type GOTO 385 where you mean GOTO 835. In fact, you could type in the line in any order and the Proofreader wouldn't notice. The Proofreader should help you catch most typing mistakes, but keep this in mind if a program that checks out with the Proofreader still seems to have errors.

Here's another thing to watch out for: If you enter the line by using abbreviations for commands, the checksum will not match up. But there is a way to make the Proofreader check it. After entering the line, LIST it. This eliminates the abbreviations. Then move the cursor up to the line and press RETURN. It should now match the checksum. You can check whole groups of lines this way.

Special Tape SAVE Instructions

When you're through typing in a listing, you must disable the Proofreader before saving the program on tape. Disable the Proofreader by pressing RUN/STOP–RESTORE (hold down the RUN/STOP key and sharply hit the RESTORE key). This procedure is not necessary for disk SAVEs, *but you must disable the Proofreader this way before a tape SAVE.*

SAVE to tape erases the Proofreader from memory, so you'll have to load and run it again if you want to type another listing. SAVE to disk does not erase the Proofreader.

Hidden Perils

The Proofreader's home in memory is not a very safe haven. Since the cassette buffer is wiped out during tape operations, you need to disable the Proofreader with RUN/STOP–RESTORE

before you save your program. This applies only to tape use. Disk users have nothing to worry about.

Not so for Commodore owners with tape drives. What if you type in a program in several sittings? The next day, you come to your computer, load and run the Proofreader, then try to load the partially completed program so you can add to it. But since the Proofreader is trying to hide in the cassette buffer, it is wiped out!

What you need is a way to load the Proofreader after you've loaded the partial program. The problem is, a tape LOAD to the buffer destroys what it's supposed to load.

If you intend to type in a program in more than one sitting or wish to make a safety SAVE, follow this procedure:

1. Load and run the Proofreader.
2. Disable it by pressing RUN/STOP–RESTORE.
3. Type the following three lines in direct mode (without line numbers):

```
A$="PROOFREADER.T":B$="{10 SPACES}":FORX=1TO4:A$=A
  $+B$:NEXTX
FORX=886TO1018:A$=A$+CHR$(PEEK(X)):NEXTX
OPEN1,1,1,A$:CLOSE1
```

After you enter the last line, you will be asked to press RECORD and PLAY on your cassette recorder. Put this program at the beginning of a new tape.

You now have a new version of the Proofreader. Turn your computer off and on, then load the program you were working on. Put the cassette containing the Proofreader into the tape unit and type:

OPEN1:CLOSE1

You will see the message "FOUND PROOFREADER.T," but not the familiar loading message. Don't worry; the Proofreader has been loaded into memory. You can now start the Proofreader by typing SYS 886. To test this, PRINT PEEK (886) should return the number 173. If it does not, repeat the steps above, making sure that A$ ("PROOFREADER.T") contains 13 characters and that B$ contains ten spaces.

You can now reload the Proofreader into memory whenever LOAD or SAVE destroys it, restoring your personal typing helper.

Program 2-3. The Automatic Proofreader

```
100 PRINT"{CLR}PLEASE WAIT...":FORI=886TO1018:READ
    A:CK=CK+A:POKEI,A:NEXT
110 IF CK<>17539 THEN PRINT"{DOWN}YOU MADE AN ERRO
    R":PRINT"IN DATA STATEMENTS.":END
120 SYS886:PRINT"{CLR}{2 DOWN}PROOFREADER ACTIVATE
    D.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
910 DATA 141,037,003,169,000,133
916 DATA 254,096,032,087,241,133
922 DATA 251,134,252,132,253,008
928 DATA 201,013,240,017,201,032
934 DATA 240,005,024,101,254,133
940 DATA 254,165,251,166,252,164
946 DATA 253,040,096,169,013,032
952 DATA 210,255,165,214,141,251
958 DATA 003,206,251,003,169,000
964 DATA 133,216,169,019,032,210
970 DATA 255,169,018,032,210,255
976 DATA 169,058,032,210,255,166
982 DATA 254,169,000,133,254,172
988 DATA 151,003,192,087,208,006
994 DATA 032,205,189,076,235,003
1000 DATA 032,205,221,169,032,032
1006 DATA 210,255,032,210,255,173
1012 DATA 251,003,133,214,076,173
1018 DATA 003
```

SpeedScript Program Listings

Before you begin typing *SpeedScript*, you must enter the certain POKEs *before* you load and run the MLX program (these POKEs are repeated below for your conveneince).

Commodore 64:

POKE 44,33:POKE 8448,0:NEW

VIC-20:

POKE 44,42:POKE 10752,0:NEW

Once the proper POKEs have been entered, load and run the MLX program. Answer the prompts for the starting address and the ending address as follows:

Commodore 64:

Starting Address? 2049
Ending Address? 8204

VIC-20:

Starting Address? 4609
Ending Address? 10482

Program 2-4. Commodore 64 *SpeedScript*

To enter this program, you must use Program 2-1, "Commodore 64 MLX," found earlier in this chapter.

```
2049 :011,008,010,000,158,050,238
2055 :048,054,049,000,000,000,158
2061 :032,136,009,169,203,205,255
2067 :110,035,141,110,035,240,178
2073 :003,032,055,009,032,197,097
2079 :009,076,105,010,165,038,178
2085 :141,067,008,165,039,141,086
2091 :068,008,165,158,141,070,141
2097 :008,165,159,141,071,008,089
2103 :166,181,240,032,169,000,075
2109 :141,021,032,160,000,185,088
2115 :000,000,153,000,000,200,164
2121 :204,021,032,208,244,238,252
2127 :068,008,238,071,008,224,184
2133 :060,240,007,202,208,224,198
2139 :165,180,208,222,096,165,103
2145 :181,170,005,180,208,001,074
```

```
2151 :096,024,138,101,039,141,130
2157 :139,008,165,038,141,138,226
2163 :008,024,138,101,159,141,174
2169 :142,008,165,158,141,141,108
2175 :008,232,164,180,208,004,155
2181 :240,013,160,255,185,000,218
2187 :000,153,000,000,136,192,108
2193 :255,208,245,206,139,008,182
2199 :206,142,008,202,208,234,127
2205 :096,169,040,133,195,133,155
2211 :020,169,004,133,196,169,086
2217 :216,133,021,173,017,032,249
2223 :133,251,173,018,032,133,147
2229 :252,162,001,173,020,032,053
2235 :133,012,173,029,013,141,176
2241 :032,208,160,000,173,044,042
2247 :013,145,020,177,251,153,190
2253 :029,032,200,041,127,201,067
2259 :031,240,019,192,040,208,173
2265 :235,136,177,251,041,127,160
2271 :201,032,240,005,136,208,021
2277 :245,160,039,200,132,059,040
2283 :136,185,029,032,145,195,189
2289 :136,016,248,164,059,024,120
2295 :152,101,251,133,251,165,020
2301 :252,105,000,133,252,224,195
2307 :001,208,003,140,016,032,147
2313 :192,040,240,008,169,032,178
2319 :145,195,200,076,009,009,137
2325 :024,165,195,105,040,133,171
2331 :195,133,020,144,004,230,241
2337 :196,230,021,232,224,025,193
2343 :240,003,076,195,008,165,214
2349 :251,141,027,032,165,252,145
2355 :141,028,032,096,173,008,017
2361 :032,133,251,141,017,032,151
2367 :141,023,032,133,057,173,110
2373 :009,032,133,252,141,018,142
2379 :032,141,024,032,133,058,239
2385 :056,173,011,032,237,009,087
2391 :032,170,169,032,160,255,137
2397 :198,252,145,251,200,230,089
2403 :252,145,251,200,208,251,126
2409 :230,252,202,208,246,145,108
2415 :251,096,133,059,132,060,074
2421 :160,000,177,059,240,006,247
2427 :032,210,255,200,208,246,250
2433 :096,032,228,255,240,251,207
2439 :096,169,147,032,210,255,020
2445 :169,054,133,001,169,000,155
```

```
2451 :141,020,032,141,008,032,009
2457 :141,010,032,141,012,032,009
2463 :141,014,032,141,176,032,183
2469 :141,207,032,169,036,024,006
2475 :105,001,141,009,032,169,116
2481 :207,141,011,032,169,208,177
2487 :141,013,032,169,255,141,166
2493 :015,032,141,174,032,076,147
2499 :132,255,032,226,013,169,254
2505 :128,141,138,002,133,157,132
2511 :032,093,017,169,006,141,153
2517 :024,003,169,010,141,025,073
2523 :003,173,008,032,133,057,113
2529 :173,009,032,133,058,032,150
2535 :246,009,169,038,160,030,115
2541 :032,113,009,238,019,032,168
2547 :096,234,234,032,078,010,159
2553 :169,018,160,030,032,113,003
2559 :009,169,000,141,019,032,113
2565 :096,072,138,072,152,072,095
2571 :169,127,141,013,221,172,086
2577 :013,221,016,003,076,114,204
2583 :254,173,113,036,240,006,077
2589 :165,002,160,000,145,057,046
2595 :169,002,133,012,032,204,075
2601 :255,032,078,010,169,247,064
2607 :160,031,032,113,009,032,168
2613 :167,016,208,009,032,069,042
2619 :020,120,169,127,076,102,161
2625 :254,032,069,020,162,250,084
2631 :154,032,197,009,076,105,132
2637 :010,162,039,169,032,157,134
2643 :000,004,202,016,250,169,212
2649 :019,076,210,255,072,041,250
2655 :128,074,133,059,104,041,122
2661 :063,005,059,096,160,000,228
2667 :140,113,036,177,057,133,251
2673 :002,160,000,177,057,073,070
2679 :128,145,057,173,113,036,003
2685 :073,001,141,113,036,032,009
2691 :158,008,032,228,255,208,252
2697 :013,165,162,041,016,240,006
2703 :245,169,000,133,162,076,160
2709 :114,010,170,160,000,165,000
2715 :002,145,057,140,113,036,136
2721 :224,095,208,012,032,112,076
2727 :012,169,032,160,000,145,173
2733 :057,076,105,010,173,019,101
2739 :032,240,007,138,072,032,188
2745 :246,009,104,170,138,201,029
```

```
2751 :013,208,002,162,095,138,041
2757 :041,127,201,032,144,078,052
2763 :224,160,208,002,162,032,223
2769 :138,072,160,000,177,057,045
2775 :201,031,240,005,173,020,117
2781 :032,240,003,032,056,016,088
2787 :104,032,093,010,160,000,114
2793 :145,057,032,158,008,056,177
2799 :165,057,237,023,032,133,118
2805 :059,165,058,237,024,032,052
2811 :005,059,144,014,165,057,183
2817 :105,000,141,023,032,165,211
2823 :058,105,000,141,024,032,111
2829 :230,057,208,002,230,058,030
2835 :032,177,011,076,105,010,174
2841 :138,174,059,011,221,059,175
2847 :011,240,006,202,208,248,178
2853 :076,105,010,202,138,010,066
2859 :170,169,010,072,169,104,225
2865 :072,189,100,011,072,189,170
2871 :099,011,072,096,039,029,145
2877 :157,137,133,002,012,138,128
2883 :134,020,148,004,019,009,145
2889 :147,135,139,005,136,140,007
2895 :022,145,017,159,018,024,208
2901 :026,016,028,030,006,001,192
2907 :011,008,031,003,131,010,029
2913 :141,007,102,012,111,012,226
2919 :122,012,176,012,016,013,198
2925 :029,013,044,013,146,013,111
2931 :217,014,055,016,013,015,189
2937 :080,015,157,016,190,016,083
2943 :224,016,001,017,163,017,053
2949 :202,019,181,018,025,020,086
2955 :044,013,146,013,097,020,216
2961 :123,021,033,022,244,012,088
2967 :179,022,168,019,079,027,133
2973 :244,014,049,022,225,013,212
2979 :232,027,239,029,244,015,181
2985 :236,015,139,028,028,016,119
2991 :199,027,032,015,012,056,004
2997 :165,057,237,017,032,165,086
3003 :058,237,018,032,176,032,228
3009 :056,173,017,032,237,008,204
3015 :032,133,059,173,018,032,134
3021 :237,009,032,005,059,240,019
3027 :013,165,057,141,017,032,124
3033 :165,058,141,018,032,032,151
3039 :158,008,056,173,027,032,165
3045 :229,057,133,251,173,028,076
```

```
3051 :032,229,058,133,252,005,176
3057 :251,240,002,176,024,024,190
3063 :173,017,032,109,016,032,114
3069 :141,017,032,173,018,032,154
3075 :105,000,141,018,032,032,075
3081 :158,008,076,225,011,096,071
3087 :056,173,023,032,237,010,034
3093 :032,133,059,173,024,032,218
3099 :237,011,032,005,059,144,003
3105 :012,173,010,032,141,023,168
3111 :032,173,011,032,141,024,196
3117 :032,056,165,057,237,008,088
3123 :032,133,059,165,058,237,223
3129 :009,032,005,059,176,011,093
3135 :173,008,032,133,057,173,127
3141 :009,032,133,058,096,056,197
3147 :165,057,237,023,032,133,210
3153 :059,165,058,237,024,032,144
3159 :005,059,176,001,096,173,085
3165 :023,032,133,057,173,024,023
3171 :032,133,058,096,230,057,193
3177 :208,002,230,058,076,177,088
3183 :011,165,057,208,002,198,240
3189 :058,198,057,076,177,011,182
3195 :165,057,133,251,165,058,184
3201 :133,252,198,252,160,255,099
3207 :177,251,201,032,240,004,016
3213 :201,031,208,003,136,208,160
3219 :243,177,251,201,032,240,011
3225 :008,201,031,240,004,136,005
3231 :208,243,096,056,152,101,247
3237 :251,133,057,165,252,105,104
3243 :000,133,058,076,177,011,114
3249 :160,000,177,057,201,032,036
3255 :240,008,201,031,240,004,139
3261 :200,208,243,096,200,208,064
3267 :011,230,058,165,058,205,154
3273 :024,032,144,002,208,025,124
3279 :177,057,201,032,240,236,126
3285 :201,031,240,232,024,152,069
3291 :101,057,133,057,165,058,022
3297 :105,000,133,058,076,177,006
3303 :011,173,023,032,133,057,148
3309 :173,024,032,133,058,076,221
3315 :177,011,169,000,141,017,246
3321 :032,173,024,032,056,233,031
3327 :004,205,009,032,176,003,172
3333 :173,009,032,141,018,032,154
3339 :032,158,008,076,232,012,017
3345 :238,029,013,173,029,013,000
```

```
3351 :041,015,141,029,013,096,102
3357 :012,238,044,013,173,044,041
3363 :013,041,015,141,044,013,046
3369 :076,158,008,011,165,057,004
3375 :133,251,165,058,133,252,015
3381 :198,252,160,255,177,251,066
3387 :201,046,240,012,201,033,024
3393 :240,008,201,063,240,004,053
3399 :201,031,208,004,136,208,091
3405 :235,096,177,251,201,046,059
3411 :240,027,201,033,240,023,079
3417 :201,063,240,019,201,031,076
3423 :240,015,136,208,235,198,103
3429 :252,165,252,205,008,032,247
3435 :176,226,076,134,013,132,096
3441 :059,198,059,200,240,010,111
3447 :177,251,201,032,240,247,243
3453 :136,076,162,012,164,059,222
3459 :076,079,013,173,008,032,000
3465 :133,057,173,009,032,133,162
3471 :058,076,177,011,160,000,113
3477 :177,057,201,046,240,029,131
3483 :201,033,240,025,201,063,150
3489 :240,021,201,031,240,017,143
3495 :200,208,235,230,058,165,239
3501 :058,205,024,032,240,226,190
3507 :144,224,076,232,012,200,043
3513 :208,014,230,058,165,058,150
3519 :205,024,032,144,005,240,073
3525 :003,076,232,012,177,057,242
3531 :201,032,240,233,201,046,132
3537 :240,229,201,033,240,225,097
3543 :201,063,240,221,201,031,148
3549 :240,217,076,217,012,173,132
3555 :012,032,141,140,032,173,245
3561 :013,032,141,141,032,032,112
3567 :078,010,169,058,160,030,232
3573 :032,113,009,169,001,141,198
3579 :019,032,096,056,165,057,164
3585 :237,008,032,133,059,165,123
3591 :058,237,009,032,005,059,151
3597 :208,003,104,104,096,165,181
3603 :057,133,038,165,058,133,091
3609 :039,096,056,165,057,133,059
3615 :158,073,255,101,038,141,029
3621 :144,032,165,058,133,159,216
3627 :073,255,101,039,141,145,029
3633 :032,165,038,141,146,032,091
3639 :165,039,141,147,032,165,232
3645 :158,141,148,032,133,038,199
```

```
3651 :165,159,141,149,032,133,078
3657 :039,056,173,145,032,109,115
3663 :141,032,205,015,032,144,136
3669 :020,032,078,010,169,073,211
3675 :160,030,032,113,009,169,092
3681 :001,141,019,032,169,000,203
3687 :133,198,096,173,140,032,107
3693 :133,158,173,141,032,133,111
3699 :159,173,144,032,133,180,168
3705 :024,109,140,032,141,140,195
3711 :032,173,145,032,133,181,055
3717 :109,141,032,141,141,032,217
3723 :169,000,141,026,208,169,084
3729 :052,133,001,032,035,008,150
3735 :169,054,133,001,169,001,166
3741 :141,026,208,173,146,032,115
3747 :133,038,173,147,032,133,051
3753 :039,173,148,032,133,158,084
3759 :173,149,032,133,159,056,109
3765 :173,023,032,229,158,133,161
3771 :180,173,024,032,229,159,216
3777 :133,181,032,035,008,056,126
3783 :173,023,032,237,144,032,072
3789 :141,023,032,173,024,032,118
3795 :237,145,032,141,024,032,054
3801 :096,032,254,013,032,112,244
3807 :012,032,027,014,056,173,025
3813 :140,032,233,001,141,140,148
3819 :032,173,141,032,233,000,078
3825 :141,141,032,096,173,141,197
3831 :002,201,005,208,003,076,230
3837 :122,015,032,103,012,032,057
3843 :254,013,032,112,012,032,202
3849 :027,014,076,227,014,032,143
3855 :226,013,169,002,133,012,058
3861 :032,078,010,169,085,160,043
3867 :030,032,113,009,032,130,117
3873 :009,072,032,246,009,104,249
3879 :041,191,201,023,208,009,200
3885 :032,254,013,032,123,012,255
3891 :076,027,014,201,019,208,084
3897 :009,032,254,013,032,045,186
3903 :013,076,027,014,201,016,154
3909 :208,009,032,254,013,032,105
3915 :002,017,076,027,014,096,051
3921 :056,165,057,237,017,032,133
3927 :133,059,165,058,237,018,245
3933 :032,005,059,240,011,173,101
3939 :017,032,133,057,173,018,017
```

```
3945 :032,133,058,096,173,008,093
3951 :032,133,057,173,009,032,035
3957 :133,058,076,177,011,165,225
3963 :057,133,251,133,158,165,252
3969 :058,133,252,133,159,160,000
3975 :000,177,251,201,032,208,236
3981 :030,200,208,247,165,252,219
3987 :205,024,032,144,015,173,228
3993 :023,032,133,251,173,024,021
3999 :032,133,252,160,000,076,044
4005 :172,015,230,252,076,136,022
4011 :015,024,152,101,251,133,079
4017 :038,169,000,101,252,133,102
4023 :039,056,173,023,032,229,223
4029 :158,133,180,173,024,032,121
4035 :229,159,133,181,056,165,094
4041 :038,229,158,141,144,032,175
4047 :165,039,229,159,141,145,061
4053 :032,032,035,008,056,173,037
4059 :023,032,237,144,032,141,060
4065 :023,032,173,024,032,237,234
4071 :145,032,141,024,032,096,189
4077 :169,255,141,169,032,076,055
4083 :007,016,169,005,141,169,238
4089 :032,032,007,016,177,057,058
4095 :201,032,208,001,200,076,205
4101 :217,012,169,000,141,170,202
4107 :032,032,078,016,169,032,114
4113 :174,169,032,160,000,145,185
4119 :057,200,202,208,250,096,012
4125 :032,056,016,032,056,016,237
4131 :169,031,160,000,145,057,085
4137 :200,145,057,032,158,008,129
4143 :032,103,012,032,103,012,085
4149 :076,245,015,169,001,141,188
4155 :169,032,169,000,141,170,228
4161 :032,032,078,016,169,032,168
4167 :160,000,145,057,076,177,174
4173 :011,024,173,023,032,109,193
4179 :169,032,173,024,032,109,110
4185 :170,032,205,011,032,144,171
4191 :005,104,104,076,157,016,045
4197 :024,165,057,133,038,109,115
4203 :169,032,133,158,165,058,054
4209 :133,039,109,170,032,133,217
4215 :159,056,173,023,032,229,023
4221 :038,133,180,173,024,032,193
4227 :229,039,133,181,032,096,073
4233 :008,024,173,023,032,109,250
4239 :169,032,141,023,032,173,201
```

```
4245 :024,032,109,170,032,141,145
4251 :024,032,096,173,020,032,020
4257 :073,014,141,020,032,096,025
4263 :169,100,160,030,032,113,003
4269 :009,032,159,255,032,228,120
4275 :255,240,248,201,147,240,230
4281 :244,041,127,201,089,096,215
4287 :169,002,133,012,032,078,105
4293 :010,169,123,160,030,032,209
4299 :113,009,032,167,016,240,012
4305 :003,076,246,009,162,250,187
4311 :154,032,055,009,032,200,185
4317 :009,076,105,010,160,000,069
4323 :177,057,201,031,240,017,182
4329 :200,208,247,230,058,165,061
4335 :058,205,024,032,144,238,172
4341 :240,236,076,232,012,200,217
4347 :208,002,230,058,076,217,018
4353 :012,165,057,133,251,165,016
4359 :058,133,252,198,252,160,036
4365 :255,177,251,201,031,240,144
4371 :017,136,192,255,208,245,048
4377 :198,252,165,252,205,009,082
4383 :032,176,236,076,134,013,186
4389 :056,152,101,251,133,251,213
4395 :169,000,101,252,133,252,182
4401 :056,165,251,229,057,133,172
4407 :059,165,252,229,058,005,055
4413 :059,208,018,132,059,024,049
4419 :165,251,229,059,133,251,131
4425 :165,252,233,000,133,252,084
4431 :076,020,017,165,251,133,229
4437 :057,165,252,133,058,076,058
4443 :177,011,120,169,000,141,197
4449 :014,220,169,027,141,017,173
4455 :208,169,124,141,020,003,000
4461 :169,017,141,021,003,169,117
4467 :001,141,026,208,141,018,138
4473 :208,088,096,169,058,164,136
4479 :012,205,018,208,208,005,015
4485 :169,001,172,029,013,140,145
4491 :033,208,141,018,208,201,180
4497 :001,240,008,169,001,141,193
4503 :025,208,076,188,254,169,047
4509 :001,141,025,208,076,049,145
4515 :234,173,141,002,041,001,243
4521 :208,003,032,226,013,032,171
4527 :078,010,169,138,160,030,248
4533 :032,113,009,160,000,177,160
4539 :057,073,128,145,057,032,167
```

```
4545 :158,008,160,000,177,057,241
4551 :073,128,145,057,169,002,005
4557 :133,012,032,130,009,009,018
4563 :064,201,087,208,009,032,044
4569 :001,018,032,177,012,076,021
4575 :016,018,201,083,208,009,246
4581 :032,001,018,032,147,013,216
4587 :076,016,018,201,080,208,066
4593 :009,032,001,018,032,225,046
4599 :016,076,016,018,032,177,070
4605 :011,076,246,009,165,057,049
4611 :133,158,141,134,032,165,254
4617 :058,133,159,141,135,032,155
4623 :096,056,165,057,133,038,048
4629 :237,134,032,141,144,032,229
4635 :165,058,133,039,237,135,026
4641 :032,141,145,032,032,050,209
4647 :014,173,134,032,133,057,070
4653 :173,135,032,133,058,032,096
4659 :158,008,076,184,017,169,151
4665 :039,229,211,141,025,032,222
4671 :160,000,169,153,032,210,019
4677 :255,169,018,032,210,255,240
4683 :169,032,032,210,255,169,174
4689 :157,032,210,255,140,026,133
4695 :032,032,130,009,172,026,232
4701 :032,133,059,169,146,032,152
4707 :210,255,169,032,032,210,239
4713 :255,169,157,032,210,255,159
4719 :169,155,032,210,255,165,073
4725 :059,201,013,240,050,201,113
4731 :020,208,015,136,016,004,010
4737 :200,076,065,018,169,157,046
4743 :032,210,255,076,065,018,023
4749 :165,059,041,127,201,032,254
4755 :144,172,204,025,032,240,196
4761 :167,165,059,153,069,032,030
4767 :032,210,255,169,000,133,190
4773 :212,133,216,200,076,065,043
4779 :018,032,210,255,169,000,087
4785 :153,069,032,152,096,032,199
4791 :078,010,169,188,160,030,050
4797 :032,113,009,032,028,019,166
4803 :176,032,173,008,032,133,237
4809 :251,173,009,032,133,252,027
4815 :174,023,032,172,024,032,152
4821 :169,251,032,216,255,176,032
4827 :009,165,144,041,191,208,209
4833 :003,076,010,020,240,039,101
4839 :173,027,019,201,008,144,035
```

```
4845 :006,032,150,027,076,005,021
4851 :019,173,027,019,201,001,171
4857 :240,249,032,078,010,169,003
4863 :194,160,030,032,113,009,025
4869 :032,093,017,169,001,141,202
4875 :019,032,096,032,078,010,022
4881 :169,205,160,030,032,113,214
4887 :009,076,005,019,000,032,164
4893 :056,018,240,022,169,236,002
4899 :160,030,032,113,009,032,155
4905 :130,009,162,008,201,068,107
4911 :240,012,162,001,201,084,235
4917 :240,006,032,246,009,104,178
4923 :104,096,142,027,019,169,104
4929 :001,160,000,032,186,255,187
4935 :160,000,224,001,240,049,233
4941 :185,069,032,201,064,234,094
4947 :234,185,070,032,201,058,095
4953 :240,035,185,071,032,201,085
4959 :058,240,028,169,048,141,011
4965 :109,032,169,058,141,110,208
4971 :032,185,069,032,153,111,177
4977 :032,200,204,026,032,144,239
4983 :244,240,242,200,076,138,235
4989 :019,185,069,032,153,109,180
4995 :032,200,204,026,032,208,065
5001 :244,140,133,032,032,078,028
5007 :010,169,069,160,032,032,103
5013 :113,009,173,133,032,162,003
5019 :109,160,032,032,189,255,164
5025 :169,013,032,210,255,076,148
5031 :069,020,032,078,010,169,033
5037 :170,160,030,032,113,009,175
5043 :032,130,009,032,093,010,229
5049 :009,128,072,173,020,032,107
5055 :240,003,032,056,016,032,058
5061 :246,009,104,076,231,010,105
5067 :056,165,057,237,008,032,246
5073 :133,059,165,058,237,009,102
5079 :032,005,059,240,004,169,212
5085 :005,133,012,032,078,010,235
5091 :169,000,160,031,032,113,220
5097 :009,032,028,019,165,012,242
5103 :201,005,240,003,032,055,007
5109 :009,169,000,166,057,164,042
5115 :058,032,213,255,144,003,188
5121 :076,229,018,142,023,032,009
5127 :140,024,032,032,231,255,209
5133 :032,078,010,169,226,160,176
5139 :030,032,113,009,076,005,028
```

```
5145 :019,032,078,010,169,006,083
5151 :160,031,032,113,009,032,152
5157 :028,019,169,001,174,008,180
5163 :032,172,009,032,032,213,021
5169 :255,165,144,041,191,240,061
5175 :210,032,078,010,169,213,255
5181 :160,030,032,113,009,076,225
5187 :005,019,120,169,000,141,009
5193 :026,208,141,032,208,141,061
5199 :033,208,169,049,141,020,187
5205 :003,169,234,141,021,003,144
5211 :169,001,141,014,220,088,212
5217 :096,169,147,032,210,255,238
5223 :169,013,032,210,255,032,046
5229 :069,020,032,148,020,169,055
5235 :013,032,210,255,169,014,040
5241 :160,031,032,113,009,032,242
5247 :228,255,201,013,208,249,001
5253 :032,093,017,076,246,009,094
5259 :032,204,255,169,001,032,064
5265 :195,255,096,032,231,255,185
5271 :169,001,162,008,160,000,139
5277 :032,186,255,169,001,162,194
5283 :043,160,031,032,189,255,105
5289 :032,192,255,176,221,162,183
5295 :001,032,198,255,032,001,182
5301 :021,032,001,021,032,001,033
5307 :021,032,001,021,240,202,192
5313 :032,204,255,032,228,255,175
5319 :201,032,208,003,032,130,037
5325 :009,162,001,032,198,255,094
5331 :032,001,021,072,032,001,114
5337 :021,168,104,170,152,160,224
5343 :055,132,001,032,205,189,069
5349 :160,054,132,001,169,032,009
5355 :032,210,255,032,001,021,018
5361 :240,006,032,210,255,076,036
5367 :238,020,169,013,032,210,161
5373 :255,076,185,020,032,207,004
5379 :255,072,165,144,041,191,103
5385 :240,006,104,104,104,076,131
5391 :139,020,104,096,162,000,024
5397 :142,136,032,142,137,032,130
5403 :142,138,032,142,139,032,140
5409 :056,177,251,233,048,144,174
5415 :042,201,010,176,038,014,008
5421 :136,032,046,137,032,014,186
5427 :136,032,046,137,032,014,192
5433 :136,032,046,137,032,014,198
5439 :136,032,046,137,032,013,203
```

```
5445 :136,032,141,136,032,200,234
5451 :208,212,230,252,076,033,062
5457 :021,248,173,136,032,013,192
5463 :137,032,240,028,056,173,241
5469 :136,032,233,001,141,136,004
5475 :032,173,137,032,233,000,194
5481 :141,137,032,238,138,032,055
5487 :208,003,238,139,032,076,039
5493 :083,021,173,138,032,216,012
5499 :096,056,173,140,032,237,089
5505 :012,032,141,142,032,173,149
5511 :141,032,237,013,032,141,219
5517 :143,032,013,142,032,208,199
5523 :016,032,078,010,169,052,248
5529 :160,031,032,113,009,169,155
5535 :001,141,019,032,096,024,216
5541 :165,057,133,038,109,142,041
5547 :032,133,158,165,058,133,082
5553 :039,109,143,032,133,159,024
5559 :056,173,023,032,229,038,222
5565 :133,180,173,024,032,229,192
5571 :039,133,181,024,101,159,064
5577 :205,011,032,144,016,032,129
5583 :078,010,169,044,160,031,187
5589 :032,113,009,169,001,141,166
5595 :019,032,096,032,096,008,246
5601 :024,173,142,032,133,180,141
5607 :109,023,032,141,023,032,079
5613 :173,143,032,133,181,109,240
5619 :024,032,141,024,032,165,149
5625 :057,133,158,165,058,133,185
5631 :159,173,012,032,133,038,034
5637 :173,013,032,133,039,169,052
5643 :000,141,026,208,169,052,095
5649 :133,001,032,035,008,169,139
5655 :054,133,001,169,001,141,010
5661 :026,208,076,177,011,160,175
5667 :000,177,057,170,200,177,048
5673 :057,136,145,057,200,138,006
5679 :145,057,096,160,000,177,170
5685 :057,041,063,240,010,201,153
5691 :027,176,006,177,057,073,063
5697 :064,145,057,076,103,012,010
5703 :133,059,041,063,006,059,176
5709 :036,059,016,002,009,128,071
5715 :112,002,009,064,133,059,206
5721 :096,005,075,066,005,058,138
5727 :001,001,001,000,001,000,099
5733 :080,027,014,015,018,141,140
5739 :175,032,138,072,152,072,236
```

```
5745 :056,173,159,032,237,161,163
5751 :032,173,160,032,237,162,147
5757 :032,144,031,173,175,032,200
5763 :032,210,255,173,141,002,176
5769 :041,001,141,032,208,208,000
5775 :246,165,145,201,127,208,211
5781 :009,238,032,208,032,084,240
5787 :025,076,120,024,104,168,160
5793 :104,170,173,175,032,096,143
5799 :032,078,010,169,164,160,012
5805 :031,076,113,009,076,120,086
5811 :024,173,029,013,141,111,158
5817 :036,169,000,133,012,141,164
5823 :032,208,141,029,013,032,134
5829 :189,255,169,004,141,170,101
5835 :032,160,007,173,141,002,206
5841 :041,001,208,003,076,104,130
5847 :023,032,078,010,169,071,086
5853 :160,031,032,113,009,032,086
5859 :130,009,041,127,162,003,187
5865 :142,170,032,201,083,240,077
5871 :086,162,008,142,170,032,071
5877 :201,068,240,034,201,080,045
5883 :208,180,032,078,010,169,160
5889 :109,160,031,032,113,009,199
5895 :032,130,009,056,233,048,003
5901 :201,004,144,160,201,080,035
5907 :176,156,141,170,032,076,002
5913 :070,023,032,078,010,169,151
5919 :145,160,031,032,113,009,009
5925 :032,056,018,240,135,172,178
5931 :026,032,169,044,153,069,024
5937 :032,200,169,087,153,069,247
5943 :032,200,140,026,032,173,146
5949 :026,032,162,069,160,032,030
5955 :032,189,255,173,170,032,150
5961 :168,201,004,144,026,201,049
5967 :008,176,022,032,078,010,149
5973 :169,124,160,031,032,113,202
5979 :009,032,130,009,056,233,048
5985 :048,168,016,003,076,177,073
5991 :022,169,001,174,170,032,159
5997 :032,186,255,032,167,022,035
6003 :169,001,032,195,255,032,031
6009 :192,255,162,001,032,201,196
6015 :255,144,003,076,120,024,237
6021 :162,000,142,151,032,142,250
6027 :150,032,142,171,032,142,040
6033 :172,032,142,112,036,189,060
6039 :090,022,157,152,032,232,068
```

```
6045 :224,012,208,245,169,255,246
6051 :141,166,032,141,164,032,071
6057 :162,004,189,101,022,157,036
6063 :030,033,202,208,247,173,044
6069 :008,032,133,251,173,009,019
6075 :032,133,252,160,000,140,136
6081 :165,032,204,164,032,240,006
6087 :006,173,152,032,141,165,100
6093 :032,177,251,016,003,076,248
6099 :098,025,201,031,240,044,082
6105 :153,110,033,200,238,165,092
6111 :032,173,165,032,205,153,215
6117 :032,144,230,140,022,032,061
6123 :177,251,201,032,240,020,132
6129 :206,165,032,136,208,244,208
6135 :172,022,032,076,008,024,069
6141 :200,177,251,201,032,240,074
6147 :001,136,140,022,032,152,230
6153 :056,101,251,133,251,165,198
6159 :252,105,000,133,252,160,149
6165 :000,173,166,032,201,255,080
6171 :208,003,032,009,025,173,221
6177 :164,032,240,003,032,049,041
6183 :025,056,046,164,032,173,023
6189 :022,032,141,021,032,169,206
6195 :110,133,253,169,033,133,114
6201 :254,032,051,029,032,066,009
6207 :025,173,166,032,205,156,052
6213 :032,144,003,032,151,024,199
6219 :056,165,251,237,023,032,071
6225 :133,059,165,252,237,024,183
6231 :032,005,059,240,056,144,111
6237 :054,173,151,032,240,011,242
6243 :169,000,141,150,032,141,220
6249 :155,032,032,151,024,173,160
6255 :170,032,201,003,208,003,216
6261 :032,130,009,032,225,255,032
6267 :240,251,169,001,032,195,243
6273 :255,032,231,255,173,111,162
6279 :036,141,029,013,162,250,254
6285 :154,032,246,009,076,105,251
6291 :010,076,190,023,056,173,163
6297 :154,032,237,166,032,168,174
6303 :136,136,240,008,048,006,221
6309 :032,084,025,136,208,250,132
6315 :173,151,032,240,017,141,157
6321 :021,032,169,111,133,253,128
6327 :169,035,133,254,032,049,087
6333 :025,032,051,029,032,084,186
6339 :025,032,084,025,032,084,221
```

```
6345 :025,238,159,032,208,003,098
6351 :238,160,032,173,158,032,232
6357 :208,050,173,170,032,201,023
6363 :003,240,043,201,008,240,186
6369 :039,056,173,159,032,237,153
6375 :161,032,173,160,032,237,002
6381 :162,032,144,024,032,204,067
6387 :255,032,078,010,169,179,198
6393 :160,031,032,113,009,032,114
6399 :130,009,032,167,022,162,009
6405 :001,032,201,255,173,150,049
6411 :032,240,017,141,021,032,238
6417 :169,110,133,253,169,034,117
6423 :133,254,032,049,025,032,036
6429 :051,029,172,155,032,140,096
6435 :166,032,136,240,008,048,153
6441 :006,032,084,025,136,208,020
6447 :250,096,169,032,172,152,150
6453 :032,140,165,032,240,006,156
6459 :032,106,022,136,208,250,045
6465 :096,172,157,032,024,152,186
6471 :109,166,032,141,166,032,205
6477 :032,084,025,136,208,250,044
6483 :096,169,013,032,106,022,009
6489 :173,112,036,240,003,032,173
6495 :106,022,096,141,168,032,148
6501 :041,127,032,071,022,174,056
6507 :173,025,221,173,025,240,196
6513 :009,202,208,248,206,165,127
6519 :032,076,190,026,202,138,015
6525 :010,170,140,167,032,169,045
6531 :025,072,169,144,072,189,034
6537 :193,025,072,189,192,025,065
6543 :072,096,056,173,167,032,227
6549 :101,251,133,251,165,252,022
6555 :105,000,133,252,076,190,143
6561 :023,177,251,201,031,240,060
6567 :001,136,140,167,032,096,227
6573 :018,087,065,076,082,084,073
6579 :066,083,078,072,070,064,100
6585 :080,063,088,077,073,071,125
6591 :074,032,026,050,026,059,202
6597 :026,069,026,079,026,089,000
6603 :026,099,026,109,026,124,101
6609 :026,158,026,006,026,022,217
6615 :026,246,025,236,025,227,232
6621 :025,183,026,224,026,041,234
6627 :026,200,169,000,141,164,159
6633 :032,076,162,025,200,032,248
6639 :019,021,141,163,032,076,179
```

```
6645 :162,025,200,032,019,021,192
6651 :141,161,032,173,139,032,161
6657 :141,162,032,076,162,025,087
6663 :200,032,019,021,141,159,067
6669 :032,173,139,032,141,160,178
6675 :032,076,162,025,200,032,034
6681 :019,021,141,154,032,076,212
6687 :162,025,169,000,141,158,174
6693 :032,200,076,162,025,169,189
6699 :010,141,112,036,200,076,106
6705 :162,025,200,169,001,141,235
6711 :171,032,076,162,025,200,209
6717 :032,019,021,141,152,032,202
6723 :076,162,025,200,032,019,069
6729 :021,141,153,032,076,162,146
6735 :025,200,032,019,021,141,005
6741 :155,032,076,162,025,200,223
6747 :032,019,021,141,156,032,236
6753 :076,162,025,200,032,019,099
6759 :021,141,157,032,076,162,180
6765 :025,172,167,032,200,152,089
6771 :072,032,151,024,104,168,154
6777 :140,167,032,096,032,151,227
6783 :026,136,140,150,032,160,003
6789 :001,177,251,153,109,034,090
6795 :200,204,150,032,144,245,090
6801 :240,243,200,076,162,025,067
6807 :200,177,251,201,031,208,195
6813 :249,096,032,151,026,136,079
6819 :140,151,032,160,001,177,056
6825 :251,153,110,035,200,204,098
6831 :151,032,144,245,240,243,206
6837 :076,162,025,032,151,026,141
6843 :076,162,025,200,177,251,054
6849 :201,061,240,007,136,173,243
6855 :168,032,076,217,023,200,147
6861 :032,019,021,072,173,168,178
6867 :032,041,127,170,104,157,074
6873 :238,032,032,162,025,076,014
6879 :145,025,200,162,008,177,172
6885 :251,041,063,201,004,240,005
6891 :009,162,001,201,020,240,100
6897 :003,076,177,022,142,027,176
6903 :019,200,177,251,201,058,129
6909 :240,003,076,177,022,200,203
6915 :177,251,201,031,240,009,144
6921 :032,071,022,153,106,032,169
6927 :076,002,027,152,056,233,049
6933 :003,162,109,160,032,032,007
6939 :189,255,032,204,255,169,107
```

```
6945 :002,032,195,255,169,002,176
6951 :174,027,019,160,000,032,195
6957 :186,255,032,055,009,169,239
6963 :000,166,057,164,058,032,016
6969 :213,255,144,003,076,177,157
6975 :022,142,023,032,140,024,190
6981 :032,104,104,162,001,032,248
6987 :201,255,076,180,023,032,074
6993 :231,255,169,000,032,189,189
6999 :255,169,015,162,008,160,088
7005 :015,032,186,255,032,192,037
7011 :255,144,011,169,015,032,213
7017 :195,255,032,231,255,076,125
7023 :246,009,032,078,010,169,143
7029 :029,160,031,032,113,009,235
7035 :032,056,018,240,022,162,141
7041 :015,032,201,255,176,223,007
7047 :169,069,160,032,032,113,198
7053 :009,169,013,032,210,255,061
7059 :032,204,255,032,231,255,132
7065 :169,000,032,189,255,169,199
7071 :015,162,008,160,015,032,039
7077 :186,255,032,192,255,176,237
7083 :186,032,078,010,162,015,142
7089 :032,198,255,032,056,018,000
7095 :032,204,255,169,015,032,122
7101 :195,255,032,231,255,169,046
7107 :001,141,019,032,096,032,004
7113 :240,027,173,176,032,240,065
7119 :022,032,147,028,032,022,234
7125 :028,173,174,032,201,255,052
7131 :240,009,032,182,028,032,230
7137 :158,008,076,211,027,076,013
7143 :246,009,173,141,002,201,235
7149 :005,208,038,032,078,010,096
7155 :169,213,160,031,032,113,193
7161 :009,032,056,018,141,176,169
7167 :032,208,003,076,246,009,061
7173 :160,000,185,069,032,153,092
7179 :177,032,200,204,026,032,170
7185 :208,244,076,246,009,165,197
7191 :057,133,251,165,058,133,052
7197 :252,169,255,141,174,032,028
7203 :160,001,162,000,173,176,195
7209 :032,240,080,189,177,032,023
7215 :032,093,010,209,251,240,114
7221 :002,162,255,200,208,011,123
7227 :230,252,165,252,205,024,163
7233 :032,240,002,176,054,232,033
7239 :236,176,032,208,224,024,203
```

```
7245 :152,101,251,133,059,165,170
7251 :252,105,000,133,060,173,038
7257 :023,032,197,059,173,024,085
7263 :032,229,060,144,024,056,128
7269 :165,059,237,176,032,133,135
7275 :057,141,173,032,165,060,223
7281 :233,000,133,058,141,174,084
7287 :032,032,177,011,096,032,243
7293 :078,010,169,223,160,031,028
7299 :032,113,009,169,001,141,084
7305 :019,032,096,173,141,002,088
7311 :201,005,208,035,032,078,190
7317 :010,169,233,160,031,032,016
7323 :113,009,032,056,018,141,012
7329 :207,032,240,014,160,000,046
7335 :185,069,032,153,208,032,078
7341 :200,204,026,032,208,244,063
7347 :076,246,009,056,165,057,020
7353 :133,158,237,173,032,133,027
7359 :059,165,058,133,159,237,234
7365 :174,032,005,059,208,101,008
7371 :169,255,141,174,032,024,230
7377 :173,176,032,101,057,133,113
7383 :038,169,000,101,058,133,202
7389 :039,056,173,023,032,229,005
7395 :158,133,180,173,024,032,159
7401 :229,159,133,181,032,035,234
7407 :008,056,173,023,032,237,000
7413 :176,032,141,023,032,173,054
7419 :024,032,233,000,141,024,193
7425 :032,173,207,032,240,041,214
7431 :141,169,032,169,000,141,147
7437 :170,032,032,078,016,160,245
7443 :000,185,208,032,032,093,057
7449 :010,145,057,200,204,207,080
7455 :032,208,242,024,165,057,247
7461 :109,207,032,133,057,165,228
7467 :058,105,000,133,058,076,217
7473 :177,011,160,000,204,021,110
7479 :032,240,032,177,253,048,069
7485 :029,032,071,022,032,208,199
7491 :029,032,106,022,173,172,089
7497 :032,240,010,169,008,032,052
7503 :106,022,169,095,032,106,097
7509 :022,200,076,053,029,096,049
7515 :140,167,032,041,127,141,227
7521 :168,032,032,071,022,201,111
7527 :067,208,027,056,173,163,029
7533 :032,237,021,032,074,056,049
7539 :237,152,032,168,169,032,137
```

```
7545 :032,106,022,136,208,250,107
7551 :172,167,032,076,086,029,177
7557 :201,069,208,017,056,173,089
7563 :153,032,237,021,032,056,158
7569 :237,152,032,168,169,032,167
7575 :076,121,029,201,085,208,103
7581 :008,173,172,032,073,001,104
7587 :141,172,032,201,035,208,184
7593 :026,140,167,032,174,159,099
7599 :032,173,160,032,160,055,019
7605 :132,001,032,205,189,160,132
7611 :054,132,001,172,167,032,233
7617 :076,086,029,174,168,032,246
7623 :189,238,032,032,106,022,050
7629 :076,086,029,174,171,032,005
7635 :240,026,133,059,041,127,069
7641 :201,065,144,018,201,091,169
7647 :176,014,170,165,059,041,080
7653 :128,073,128,074,074,133,071
7659 :059,138,005,059,096,032,112
7665 :078,010,056,173,010,032,088
7671 :237,023,032,170,173,011,125
7677 :032,237,024,032,160,055,025
7683 :132,001,032,205,189,160,210
7689 :054,132,001,169,001,141,251
7695 :019,032,096,008,014,155,083
7701 :146,211,080,069,069,068,152
7707 :211,067,082,073,080,084,112
7713 :032,051,046,049,000,032,243
7719 :066,089,032,195,072,065,046
7725 :082,076,069,083,032,194,069
7731 :082,065,078,078,079,078,255
7737 :000,194,085,070,070,069,033
7743 :082,032,195,076,069,065,070
7749 :082,069,068,000,194,085,055
7755 :070,070,069,082,032,198,084
7761 :085,076,076,000,196,069,071
7767 :076,069,084,069,032,040,201
7773 :211,044,215,044,208,041,088
7779 :000,058,032,193,082,069,021
7785 :032,089,079,085,032,083,249
7791 :085,082,069,063,032,040,226
7797 :217,047,206,041,058,000,174
7803 :197,210,193,211,197,032,139
7809 :193,204,204,032,212,197,147
7815 :216,212,000,197,082,065,139
7821 :083,069,032,040,211,044,108
7827 :215,044,208,041,058,032,233
7833 :018,210,197,212,213,210,189
7839 :206,146,032,084,079,032,226
```

```
7845 :069,088,073,084,000,208,175
7851 :082,069,083,083,032,070,078
7857 :079,082,077,065,084,032,084
7863 :075,069,089,058,000,211,173
7869 :065,086,069,058,000,212,167
7875 :065,080,069,032,197,210,080
7881 :210,207,210,000,211,084,099
7887 :079,080,080,069,068,000,071
7893 :214,069,082,073,070,089,042
7899 :032,197,082,082,079,082,005
7905 :000,206,079,032,069,082,181
7911 :082,079,082,083,000,147,192
7917 :032,018,212,146,065,080,022
7923 :069,032,079,082,032,018,043
7929 :196,146,073,083,075,063,117
7935 :000,204,079,065,068,058,217
7941 :000,214,069,082,073,070,001
7947 :089,058,000,208,082,069,005
7953 :083,083,032,018,210,197,128
7959 :212,213,210,206,146,000,242
7965 :196,073,083,075,032,067,043
7971 :079,077,077,065,078,068,223
7977 :058,000,036,206,079,032,196
7983 :210,079,079,077,000,206,186
7989 :079,032,084,069,088,084,233
7995 :032,073,078,032,066,085,169
8001 :070,070,069,082,046,000,146
8007 :147,208,082,073,078,084,231
8013 :032,084,079,058,032,018,124
8019 :211,146,067,082,069,069,215
8025 :078,044,018,196,146,073,132
8031 :083,075,044,018,208,146,157
8037 :082,073,078,084,069,082,057
8043 :063,000,196,069,086,073,082
8049 :067,069,032,078,085,077,009
8055 :066,069,082,063,000,211,098
8061 :069,067,079,078,068,065,039
8067 :082,089,032,193,068,068,151
8073 :082,069,083,083,032,035,009
8079 :063,000,208,082,073,078,135
8085 :084,032,084,079,032,070,018
8091 :073,076,069,078,065,077,081
8097 :069,058,000,147,208,082,213
8103 :073,078,084,073,078,071,112
8109 :046,046,046,013,013,000,081
8115 :201,078,083,069,082,084,008
8121 :032,078,069,088,084,032,056
8127 :083,072,069,069,084,044,100
8133 :032,080,082,069,083,083,114
8139 :032,018,210,197,212,213,061
```

```
8145 :210,206,146,000,200,085,032
8151 :078,084,032,070,079,082,128
8157 :058,000,206,079,084,032,168
8163 :198,079,085,078,068,000,223
8169 :210,069,080,076,065,067,032
8175 :069,032,087,073,084,072,144
8181 :058,000,197,216,201,212,105
8187 :032,211,080,069,069,068,012
8193 :211,067,082,073,080,084,086
8199 :000,013,013,013,013,013,072
```

Program 2-5. VIC-20 *SpeedScript*

To enter this program, you must use Program 2-2, "VIC-20 MLX," found earlier in this chapter and at least 8K expansion memory.

```
4609 :011,018,010,000,158,052,250
4615 :054,050,049,000,000,000,160
4621 :032,131,019,169,203,205,004
4627 :109,044,141,109,044,240,194
4633 :003,032,050,019,032,195,100
4639 :019,076,038,020,165,038,131
4645 :141,067,018,165,039,141,096
4651 :068,018,165,158,141,070,151
4657 :018,165,159,141,071,018,109
4663 :166,181,240,032,169,000,075
4669 :141,000,041,160,000,185,076
4675 :000,000,153,000,000,200,164
4681 :204,000,041,208,244,238,240
4687 :068,018,238,071,018,224,204
4693 :000,240,007,202,208,224,198
4699 :165,180,208,222,096,165,103
4705 :181,170,005,180,208,001,074
4711 :096,024,138,101,039,141,130
4717 :139,018,165,038,141,138,236
4723 :018,024,138,101,159,141,184
4729 :142,018,165,158,141,141,118
4735 :018,232,164,180,208,004,165
4741 :240,013,160,255,185,000,218
4747 :000,153,000,000,136,192,108
4753 :255,208,245,206,139,018,192
4759 :206,142,018,202,208,234,137
4765 :096,169,044,133,195,133,159
4771 :020,169,016,133,196,169,098
4777 :148,133,021,173,252,040,168
4783 :133,251,173,253,040,133,134
4789 :252,173,255,040,032,014,179
4795 :020,162,002,160,000,173,192
4801 :020,023,145,020,177,251,061
4807 :153,008,041,200,041,127,001
```

```
4813 :201,031,240,019,192,022,142
4819 :208,235,136,177,251,041,235
4825 :127,201,032,240,005,136,190
4831 :208,245,160,021,200,132,165
4837 :059,136,185,008,041,145,035
4843 :195,136,016,248,164,059,029
4849 :024,152,101,251,133,251,129
4855 :165,252,105,000,133,252,130
4861 :224,002,208,003,140,251,057
4867 :040,192,022,240,008,169,162
4873 :032,145,195,200,076,004,149
4879 :019,024,165,195,105,022,033
4885 :133,195,133,020,144,004,138
4891 :230,196,230,021,232,224,136
4897 :023,240,003,076,190,018,071
4903 :165,251,141,006,041,165,040
4909 :252,141,007,041,096,173,243
4915 :243,040,133,251,141,252,087
4921 :040,141,002,041,133,057,215
4927 :173,244,040,133,252,141,022
4933 :253,040,141,003,041,133,168
4939 :058,056,173,246,040,237,117
4945 :244,040,170,169,032,160,128
4951 :255,198,252,145,251,200,108
4957 :230,252,145,251,200,208,099
4963 :251,230,252,202,208,246,208
4969 :145,251,096,133,059,132,153
4975 :060,160,000,177,059,240,039
4981 :006,032,210,255,200,208,004
4987 :246,096,032,228,255,240,196
4993 :251,096,169,000,141,255,017
4999 :040,141,243,040,141,245,217
5005 :040,141,247,040,141,249,231
5011 :040,141,155,041,141,196,093
5017 :041,169,045,024,105,001,026
5023 :141,244,040,056,165,056,093
5029 :233,001,141,250,040,056,118
5035 :233,004,141,248,040,056,125
5041 :233,001,141,246,040,169,239
5047 :255,141,153,041,032,202,239
5053 :023,169,147,076,210,255,045
5059 :169,128,141,138,002,133,138
5065 :157,173,005,023,032,241,064
5071 :022,173,243,040,133,057,107
5077 :173,244,040,133,058,032,125
5083 :234,019,169,072,160,039,144
5089 :032,108,019,238,254,040,148
5095 :076,134,021,032,250,019,251
5101 :169,054,160,039,032,108,031
5107 :019,169,000,141,254,040,098
```

```
5113 :096,162,043,169,160,157,012
5119 :000,016,202,016,250,169,140
5125 :019,032,210,255,169,018,196
5131 :076,210,255,141,134,002,061
5137 :162,043,157,000,148,202,217
5143 :016,250,096,072,041,128,114
5149 :074,133,059,104,041,063,247
5155 :005,059,096,160,000,177,020
5161 :057,133,002,160,000,177,058
5167 :057,073,128,145,057,032,027
5173 :158,018,173,141,002,041,074
5179 :004,240,009,165,197,201,107
5185 :064,240,003,076,216,020,172
5191 :032,228,255,208,013,165,204
5197 :162,041,016,240,229,169,166
5203 :000,133,162,076,044,020,006
5209 :170,160,000,165,002,145,219
5215 :057,224,095,208,012,032,211
5221 :069,022,169,032,160,000,041
5227 :145,057,076,038,020,173,104
5233 :254,040,240,007,138,072,096
5239 :032,234,019,104,170,138,048
5245 :201,013,208,002,162,095,038
5251 :138,041,127,201,032,144,046
5257 :100,224,160,208,002,162,225
5263 :032,138,072,160,000,177,210
5269 :057,201,031,240,005,173,088
5275 :255,040,240,003,032,015,228
5281 :026,104,032,026,020,160,017
5287 :000,145,057,032,158,018,065
5293 :056,165,057,237,002,041,219
5299 :133,059,165,058,237,003,066
5305 :041,005,059,144,014,165,101
5311 :057,105,000,141,002,041,025
5317 :165,058,105,000,141,003,157
5323 :041,230,057,208,002,230,203
5329 :058,032,134,021,076,038,056
5335 :020,160,000,165,002,145,195
5341 :057,024,165,197,105,064,065
5347 :170,132,162,165,162,201,195
5353 :010,208,250,132,198,138,145
5359 :174,016,021,221,016,021,196
5365 :240,006,202,208,248,076,201
5371 :038,020,202,138,010,170,061
5377 :169,020,072,169,037,072,028
5383 :189,057,021,072,189,056,079
5389 :021,072,096,039,029,157,171
5395 :137,133,099,085,138,134,233
5401 :020,148,082,019,076,147,005
5407 :135,139,113,136,140,091,017
```

```
5413 :145,017,121,074,090,097,069
5419 :077,070,118,072,081,108,057
5425 :107,110,003,131,084,141,113
5431 :083,059,022,068,022,079,132
5437 :022,133,022,229,022,005,238
5443 :023,020,023,122,023,175,197
5449 :024,014,026,227,024,039,171
5455 :025,116,026,146,026,181,087
5461 :026,214,026,049,027,063,234
5467 :029,048,028,148,029,020,137
5473 :023,122,023,191,029,203,176
5479 :030,095,031,201,022,235,205
5485 :031,029,029,131,036,202,055
5491 :024,111,031,201,023,028,021
5497 :037,027,039,203,025,195,135
5503 :025,191,037,243,025,251,131
5509 :036,032,228,021,056,165,159
5515 :057,237,252,040,165,058,180
5521 :237,253,040,176,032,056,171
5527 :173,252,040,237,243,040,112
5533 :133,059,173,253,040,237,028
5539 :244,040,005,059,240,013,252
5545 :165,057,141,252,040,165,221
5551 :058,141,253,040,032,158,089
5557 :018,056,173,006,041,229,192
5563 :057,133,251,173,007,041,081
5569 :229,058,133,252,005,251,097
5575 :240,002,176,024,024,173,070
5581 :252,040,109,251,040,141,014
5587 :252,040,173,253,040,105,050
5593 :000,141,253,040,032,158,073
5599 :018,076,182,021,096,056,160
5605 :173,002,041,237,245,040,199
5611 :133,059,173,003,041,237,113
5617 :246,040,005,059,144,012,235
5623 :173,245,040,141,002,041,121
5629 :173,246,040,141,003,041,129
5635 :056,165,057,237,243,040,033
5641 :133,059,165,058,237,244,137
5647 :040,005,059,176,011,173,223
5653 :243,040,133,057,173,244,143
5659 :040,133,058,096,056,165,063
5665 :057,237,002,041,133,059,050
5671 :165,058,237,003,041,005,036
5677 :059,176,001,096,173,002,040
5683 :041,133,057,173,003,041,243
5689 :133,058,096,230,057,208,071
5695 :002,230,058,076,134,021,072
5701 :165,057,208,002,198,058,245
5707 :198,057,076,134,021,165,214
```

```
5713 :057,133,251,165,058,133,110
5719 :252,198,252,160,255,177,101
5725 :251,201,032,240,004,201,254
5731 :031,208,003,136,208,243,160
5737 :177,251,201,032,240,008,246
5743 :201,031,240,004,136,208,163
5749 :243,096,056,152,101,251,248
5755 :133,057,165,252,105,000,067
5761 :133,058,076,134,021,160,199
5767 :000,177,057,201,032,240,074
5773 :008,201,031,240,004,200,057
5779 :208,243,096,200,208,011,089
5785 :230,058,165,058,205,003,104
5791 :041,144,002,208,025,177,244
5797 :057,201,032,240,236,201,108
5803 :031,240,232,024,152,101,183
5809 :057,133,057,165,058,105,240
5815 :000,133,058,076,134,021,093
5821 :173,002,041,133,057,173,000
5827 :003,041,133,058,076,134,128
5833 :021,169,000,141,252,040,056
5839 :173,003,041,056,233,004,205
5845 :205,244,040,176,003,173,030
5851 :244,040,141,253,040,032,201
5857 :158,018,076,189,022,238,158
5863 :005,023,173,005,023,041,245
5869 :015,141,005,023,010,010,185
5875 :010,010,133,059,173,005,121
5881 :023,041,007,024,105,008,201
5887 :101,059,141,015,144,096,043
5893 :001,238,020,023,173,020,224
5899 :023,041,007,141,020,023,010
5905 :076,158,018,000,165,057,235
5911 :133,251,165,058,133,252,247
5917 :198,252,160,255,177,251,042
5923 :201,046,240,012,201,033,000
5929 :240,008,201,063,240,004,029
5935 :201,031,208,004,136,208,067
5941 :235,096,177,251,201,046,035
5947 :240,027,201,033,240,023,055
5953 :201,063,240,019,201,031,052
5959 :240,015,136,208,235,198,079
5965 :252,165,252,205,243,040,210
5971 :176,226,076,110,023,132,058
5977 :059,198,059,200,240,010,087
5983 :177,251,201,032,240,247,219
5989 :136,076,119,022,164,059,165
5995 :076,055,023,173,243,040,205
6001 :133,057,173,244,040,133,125
```

```
6007 :058,076,134,021,160,000,056
6013 :177,057,201,046,240,029,107
6019 :201,033,240,025,201,063,126
6025 :240,021,201,031,240,017,119
6031 :200,208,235,230,058,165,215
6037 :058,205,003,041,240,226,154
6043 :144,224,076,189,022,200,242
6049 :208,014,230,058,165,058,126
6055 :205,003,041,144,005,240,037
6061 :003,076,189,022,177,057,185
6067 :201,032,240,233,201,046,108
6073 :240,229,201,033,240,225,073
6079 :201,063,240,221,201,031,124
6085 :240,217,076,174,022,173,075
6091 :247,040,141,119,041,173,196
6097 :248,040,141,120,041,032,063
6103 :250,019,169,093,160,039,177
6109 :032,108,019,169,001,141,179
6115 :254,040,096,056,165,057,127
6121 :237,243,040,133,059,165,086
6127 :058,237,244,040,005,059,114
6133 :208,003,104,104,096,165,157
6139 :057,133,038,165,058,133,067
6145 :039,096,056,165,057,133,035
6151 :158,073,255,101,038,141,005
6157 :123,041,165,058,133,159,180
6163 :073,255,101,039,141,124,240
6169 :041,165,038,141,125,041,064
6175 :165,039,141,126,041,165,196
6181 :158,141,127,041,133,038,163
6187 :165,159,141,128,041,133,042
6193 :039,056,173,124,041,109,079
6199 :120,041,205,250,040,144,087
6205 :020,032,250,019,169,108,147
6211 :160,039,032,108,019,169,082
6217 :001,141,254,040,169,000,166
6223 :133,198,096,173,119,041,071
6229 :133,158,173,120,041,133,075
6235 :159,173,123,041,133,180,132
6241 :024,109,119,041,141,119,138
6247 :041,173,124,041,133,181,028
6253 :109,120,041,141,120,041,169
6259 :032,035,018,173,125,041,027
6265 :133,038,173,126,041,133,253
6271 :039,173,127,041,133,158,030
6277 :173,128,041,133,159,056,055
6283 :173,002,041,229,158,133,107
6289 :180,173,003,041,229,159,162
6295 :133,181,032,035,018,056,094
6301 :173,002,041,237,123,041,006
```

```
6307 :141,002,041,173,003,041,052
6313 :237,124,041,141,003,041,244
6319 :096,032,230,023,032,069,145
6325 :022,032,003,024,056,173,235
6331 :119,041,233,001,141,119,073
6337 :041,173,120,041,233,000,033
6343 :141,120,041,096,173,141,143
6349 :002,201,005,208,003,076,188
6355 :081,025,032,060,022,032,207
6361 :230,023,032,069,022,032,113
6367 :003,024,076,185,024,032,055
6373 :202,023,169,002,032,014,159
6379 :020,032,250,019,169,120,077
6385 :160,039,032,108,019,032,119
6391 :125,019,072,032,234,019,236
6397 :104,041,191,201,023,208,253
6403 :009,032,230,023,032,080,153
6409 :022,076,003,024,201,019,098
6415 :208,009,032,230,023,032,037
6421 :021,023,076,003,024,201,113
6427 :016,208,009,032,230,023,033
6433 :032,215,026,076,003,024,153
6439 :096,056,165,057,237,252,134
6445 :040,133,059,165,058,237,225
6451 :253,040,005,059,240,011,147
6457 :173,252,040,133,057,173,117
6463 :253,040,133,058,096,173,048
6469 :243,040,133,057,173,244,191
6475 :040,133,058,076,134,021,025
6481 :165,057,133,251,133,158,210
6487 :165,058,133,252,133,159,219
6493 :160,000,177,251,201,032,146
6499 :208,030,200,208,247,165,133
6505 :252,205,003,041,144,015,253
6511 :173,002,041,133,251,173,116
6517 :003,041,133,252,160,000,194
6523 :076,131,025,230,252,076,145
6529 :095,025,024,152,101,251,009
6535 :133,038,169,000,101,252,060
6541 :133,039,056,173,002,041,073
6547 :229,158,133,180,173,003,255
6553 :041,229,159,133,181,056,184
6559 :165,038,229,158,141,123,245
6565 :041,165,039,229,159,141,171
6571 :124,041,032,035,018,056,221
6577 :173,002,041,237,123,041,026
6583 :141,002,041,173,003,041,072
6589 :237,124,041,141,003,041,008
6595 :096,169,255,141,148,041,021
6601 :076,222,025,169,005,141,071
```

```
6607 :148,041,032,222,025,177,084
6613 :057,201,032,208,001,200,144
6619 :076,174,022,169,000,141,033
6625 :149,041,032,037,026,169,167
6631 :032,174,148,041,160,000,018
6637 :145,057,200,202,208,250,019
6643 :096,032,015,026,032,015,203
6649 :026,169,031,160,000,145,012
6655 :057,200,145,057,032,158,136
6661 :018,032,060,022,032,060,229
6667 :022,076,204,025,169,001,252
6673 :141,148,041,169,000,141,145
6679 :149,041,032,037,026,169,221
6685 :032,160,000,145,057,076,243
6691 :134,021,024,173,002,041,174
6697 :109,148,041,173,003,041,044
6703 :109,149,041,205,246,040,069
6709 :144,005,104,104,076,116,090
6715 :026,024,165,057,133,038,246
6721 :109,148,041,133,158,165,051
6727 :058,133,039,109,149,041,088
6733 :133,159,056,173,002,041,129
6739 :229,038,133,180,173,003,071
6745 :041,229,039,133,181,032,232
6751 :096,018,024,173,002,041,193
6757 :109,148,041,141,002,041,071
6763 :173,003,041,109,149,041,111
6769 :141,003,041,096,173,255,054
6775 :040,073,006,141,255,040,162
6781 :096,169,135,160,039,032,244
6787 :108,019,032,228,255,240,245
6793 :251,201,147,240,247,041,240
6799 :127,201,089,096,169,002,059
6805 :032,014,020,032,250,019,004
6811 :169,148,160,039,032,108,043
6817 :019,032,126,026,240,003,095
6823 :076,234,019,162,250,154,038
6829 :032,050,019,032,195,019,008
6835 :076,038,020,160,000,177,138
6841 :057,201,031,240,017,200,163
6847 :208,247,230,058,165,058,133
6853 :205,003,041,144,238,240,044
6859 :236,076,189,022,200,208,110
6865 :002,230,058,076,174,022,003
6871 :165,057,133,251,165,058,020
6877 :133,252,198,252,160,255,191
6883 :177,251,201,031,240,017,120
6889 :136,192,255,208,245,198,187
6895 :252,165,252,205,244,040,117
6901 :176,236,076,110,023,056,154
```

```
6907 :152,101,251,133,251,169,028
6913 :000,101,252,133,252,056,027
6919 :165,251,229,057,133,059,133
6925 :165,252,229,058,005,059,013
6931 :208,018,132,059,024,165,113
6937 :251,229,059,133,251,165,089
6943 :252,233,000,133,252,076,209
6949 :233,026,165,251,133,057,134
6955 :165,252,133,058,076,134,093
6961 :021,173,141,002,041,001,172
6967 :208,003,032,202,023,032,043
6973 :250,019,169,158,160,039,088
6979 :032,108,019,160,000,177,051
6985 :057,073,128,145,057,032,053
6991 :158,018,160,000,177,057,137
6997 :073,128,145,057,169,002,147
7003 :032,014,020,032,125,019,077
7009 :009,064,201,087,208,009,163
7015 :032,144,027,032,134,022,238
7021 :076,159,027,201,083,208,095
7027 :009,032,144,027,032,123,226
7033 :023,076,159,027,201,080,175
7039 :208,009,032,144,027,032,067
7045 :182,026,076,159,027,032,123
7051 :134,021,076,234,019,165,020
7057 :057,133,158,141,113,041,020
7063 :165,058,133,159,141,114,153
7069 :041,096,056,165,057,133,193
7075 :038,237,113,041,141,123,088
7081 :041,165,058,133,039,237,074
7087 :114,041,141,124,041,032,156
7093 :026,024,173,113,041,133,179
7099 :057,173,114,041,133,058,251
7105 :032,158,018,076,070,027,062
7111 :169,044,229,211,141,004,229
7117 :041,160,000,169,166,032,005
7123 :210,255,169,157,032,210,220
7129 :255,140,005,041,032,125,047
7135 :019,172,005,041,133,059,140
7141 :169,032,032,210,255,169,072
7147 :157,032,210,255,165,059,089
7153 :201,013,240,050,201,020,198
7159 :208,015,136,016,004,200,058
7165 :076,208,027,169,157,032,154
7171 :210,255,076,208,027,165,176
7177 :059,041,127,201,032,144,101
7183 :192,204,004,041,240,187,115
7189 :165,059,153,048,041,032,007
7195 :210,255,169,000,133,212,238
7201 :133,216,200,076,208,027,125
```

```
7207 :032,210,255,169,000,153,090
7213 :048,041,152,096,032,250,152
7219 :019,169,214,160,039,032,172
7225 :108,019,032,148,028,176,056
7231 :032,173,243,040,133,251,167
7237 :173,244,040,133,252,174,061
7243 :002,041,172,003,041,169,247
7249 :251,032,216,255,176,009,252
7255 :165,144,041,191,208,003,071
7261 :076,130,029,240,036,173,009
7267 :147,028,201,008,144,006,121
7273 :032,202,036,076,128,028,095
7279 :173,147,028,201,001,240,133
7285 :249,032,250,019,169,220,032
7291 :160,039,032,108,019,169,138
7297 :001,141,254,040,096,032,181
7303 :250,019,169,231,160,039,235
7309 :032,108,019,076,128,028,020
7315 :000,032,199,027,240,022,155
7321 :169,006,160,040,032,108,156
7327 :019,032,125,019,162,008,012
7333 :201,068,240,012,162,001,081
7339 :201,084,240,006,032,234,200
7345 :019,104,104,096,142,147,021
7351 :028,169,001,160,000,032,061
7357 :186,255,160,000,224,001,247
7363 :240,049,185,048,041,201,191
7369 :064,208,014,185,049,041,250
7375 :201,058,240,035,185,050,208
7381 :041,201,058,240,028,169,182
7387 :048,141,088,041,169,058,252
7393 :141,089,041,185,048,041,002
7399 :153,090,041,200,204,005,156
7405 :041,144,244,240,242,200,068
7411 :076,002,029,185,048,041,112
7417 :153,088,041,200,204,005,172
7423 :041,208,244,140,112,041,017
7429 :032,250,019,169,048,160,171
7435 :041,032,108,019,173,112,240
7441 :041,162,088,160,041,032,029
7447 :189,255,169,013,076,210,167
7453 :255,032,250,019,169,196,182
7459 :160,039,032,108,019,032,169
7465 :125,019,032,026,020,009,016
7471 :128,072,173,255,040,240,187
7477 :003,032,015,026,032,234,139
7483 :019,104,076,166,020,056,244
7489 :165,057,237,243,040,133,172
7495 :251,165,058,237,244,040,042
7501 :005,251,240,007,169,005,242
```

```
7507 :133,251,032,014,020,032,053
7513 :250,019,169,026,160,040,241
7519 :032,108,019,032,148,028,206
7525 :165,251,201,005,240,003,198
7531 :032,050,019,169,000,166,031
7537 :057,164,058,032,213,255,124
7543 :144,003,076,096,028,142,096
7549 :002,041,140,003,041,032,128
7555 :234,251,032,231,255,032,142
7561 :250,019,169,252,160,039,002
7567 :032,108,019,076,128,028,022
7573 :032,250,019,169,032,160,043
7579 :040,032,108,019,032,148,022
7585 :028,169,001,174,243,040,048
7591 :172,244,040,032,213,255,099
7597 :165,144,041,191,240,207,137
7603 :032,250,019,169,239,160,024
7609 :039,032,108,019,076,128,075
7615 :028,169,147,032,210,255,008
7621 :169,013,032,210,255,032,140
7627 :236,029,169,013,032,210,124
7633 :255,169,040,160,040,032,137
7639 :108,019,032,228,255,201,034
7645 :013,208,249,076,234,019,252
7651 :032,204,255,169,001,032,152
7657 :195,255,096,032,231,255,017
7663 :169,001,162,008,160,000,227
7669 :032,186,255,169,001,162,026
7675 :069,160,040,032,189,255,228
7681 :032,192,255,176,221,162,015
7687 :001,032,198,255,032,081,094
7693 :030,032,081,030,032,081,043
7699 :030,032,081,030,240,202,122
7705 :032,204,255,032,228,255,007
7711 :201,032,208,003,032,125,120
7717 :019,162,001,032,198,255,192
7723 :032,081,030,072,032,081,115
7729 :030,168,104,170,152,032,193
7735 :205,221,169,032,032,210,156
7741 :255,032,081,030,240,006,193
7747 :032,210,255,076,062,030,220
7753 :169,013,032,210,255,076,060
7759 :017,030,032,207,255,072,180
7765 :165,144,041,191,240,006,104
7771 :104,104,104,076,227,029,223
7777 :104,096,162,000,142,115,204
7783 :041,142,116,041,142,117,190
7789 :041,142,118,041,056,177,172
7795 :251,233,048,144,042,201,010
7801 :010,176,038,014,115,041,003
```

```
7807 :046,116,041,014,115,041,244
7813 :046,116,041,014,115,041,250
7819 :046,116,041,014,115,041,000
7825 :046,116,041,013,115,041,005
7831 :141,115,041,200,208,212,044
7837 :230,252,076,113,030,248,082
7843 :173,115,041,013,116,041,150
7849 :240,028,056,173,115,041,054
7855 :233,001,141,115,041,173,111
7861 :116,041,233,000,141,116,060
7867 :041,238,117,041,208,003,067
7873 :238,118,041,076,163,030,091
7879 :173,117,041,216,096,056,130
7885 :173,119,041,237,247,040,038
7891 :141,121,041,173,120,041,080
7897 :237,248,040,141,122,041,022
7903 :013,121,041,208,016,032,142
7909 :250,019,169,078,160,040,177
7915 :032,108,019,169,001,141,193
7921 :254,040,096,024,165,057,109
7927 :133,038,109,121,041,133,054
7933 :158,165,058,133,039,109,147
7939 :122,041,133,159,056,173,175
7945 :002,041,229,038,133,180,120
7951 :173,003,041,229,039,133,121
7957 :181,024,101,159,205,246,169
7963 :040,144,016,032,250,019,016
7969 :169,070,160,040,032,108,100
7975 :019,169,001,141,254,040,151
7981 :096,032,096,018,024,173,228
7987 :121,041,133,180,109,002,125
7993 :041,141,002,041,173,122,065
7999 :041,133,181,109,003,041,059
8005 :141,003,041,165,057,133,097
8011 :158,165,058,133,159,173,153
8017 :247,040,133,038,173,248,192
8023 :040,133,039,032,035,018,128
8029 :076,134,021,160,000,177,149
8035 :057,170,200,177,057,136,128
8041 :145,057,200,138,145,057,079
8047 :096,160,000,177,057,041,130
8053 :063,240,010,201,027,176,066
8059 :006,177,057,073,064,145,133
8065 :057,076,060,022,133,059,024
8071 :041,063,006,059,036,059,143
8077 :016,002,009,128,112,002,154
8083 :009,064,133,059,096,005,001
8089 :075,066,005,058,001,001,103
8095 :001,000,001,000,080,027,012
8101 :014,015,018,141,154,041,036
```

```
8107 :138,072,152,072,056,173,066
8113 :138,041,237,140,041,173,179
8119 :139,041,237,141,041,144,158
8125 :025,173,154,041,032,210,056
8131 :255,173,141,002,041,001,040
8137 :208,249,165,145,201,127,016
8143 :208,006,032,136,034,076,187
8149 :168,033,104,168,104,170,192
8155 :173,154,041,096,032,250,197
8161 :019,169,173,160,040,076,094
8167 :108,019,076,168,033,169,036
8173 :000,032,189,255,173,020,138
8179 :023,141,134,002,169,004,204
8185 :141,149,041,160,007,173,152
8191 :141,002,041,001,208,003,139
8197 :076,152,032,032,250,019,054
8203 :169,097,160,040,032,108,105
8209 :019,032,125,019,041,127,124
8215 :162,003,142,149,041,201,209
8221 :083,240,086,162,008,142,238
8227 :149,041,201,068,240,034,000
8233 :201,080,208,188,032,250,232
8239 :019,169,127,160,040,032,082
8245 :108,019,032,125,019,056,156
8251 :233,048,201,004,144,168,089
8257 :201,080,176,164,141,149,208
8263 :041,076,118,032,032,250,108
8269 :019,169,163,160,040,032,148
8275 :108,019,032,199,027,240,196
8281 :143,172,005,041,169,044,151
8287 :153,048,041,200,169,087,025
8293 :153,048,041,200,140,005,176
8299 :041,173,005,041,162,048,065
8305 :160,041,032,189,255,173,195
8311 :149,041,168,201,004,144,058
8317 :026,201,008,176,022,032,078
8323 :250,019,169,142,160,040,143
8329 :032,108,019,032,125,019,216
8335 :056,233,048,168,016,003,155
8341 :076,233,031,169,001,174,065
8347 :149,041,032,186,255,032,082
8353 :223,031,169,001,032,195,044
8359 :255,032,192,255,162,001,040
8365 :032,201,255,144,003,076,116
8371 :168,033,162,000,142,130,046
8377 :041,142,129,041,142,150,062
8383 :041,142,151,041,142,110,050
8389 :045,189,152,031,157,131,134
8395 :041,232,224,012,208,245,141
8401 :169,255,141,145,041,141,077
```

```
8407 :143,041,162,004,189,163,149
8413 :031,157,029,042,202,208,122
8419 :247,173,243,040,133,251,034
8425 :173,244,040,133,252,160,211
8431 :000,140,144,041,204,143,143
8437 :041,240,006,173,131,041,109
8443 :141,144,041,177,251,016,253
8449 :003,076,150,034,201,031,240
8455 :240,044,153,109,042,200,027
8461 :238,144,041,173,144,041,026
8467 :205,132,041,144,230,140,143
8473 :001,041,177,251,201,032,216
8479 :240,020,206,144,041,136,050
8485 :208,244,172,001,041,076,011
8491 :056,033,200,177,251,201,193
8497 :032,240,001,136,140,001,087
8503 :041,152,056,101,251,133,021
8509 :251,165,252,105,000,133,199
8515 :252,160,000,173,145,041,070
8521 :201,255,208,003,032,057,061
8527 :034,173,143,041,240,003,201
8533 :032,101,034,056,046,143,241
8539 :041,173,001,041,141,000,232
8545 :041,169,109,133,253,169,203
8551 :042,133,254,032,103,038,193
8557 :032,118,034,173,145,041,140
8563 :205,135,041,144,003,032,163
8569 :199,033,056,165,251,237,038
8575 :002,041,133,059,165,252,011
8581 :237,003,041,005,059,240,206
8587 :056,144,054,173,130,041,225
8593 :240,011,169,000,141,129,067
8599 :041,141,134,041,032,199,227
8605 :033,173,149,041,201,003,245
8611 :208,003,032,125,019,032,070
8617 :225,255,240,251,173,255,032
8623 :040,141,134,002,169,001,150
8629 :032,195,255,032,231,255,157
8635 :162,250,154,032,234,019,014
8641 :076,038,020,076,238,032,161
8647 :056,173,133,041,237,145,216
8653 :041,168,136,136,240,008,166
8659 :048,006,032,136,034,136,091
8665 :208,250,173,130,041,240,235
8671 :017,141,000,041,169,110,189
8677 :133,253,169,044,133,254,191
8683 :032,101,034,032,103,038,063
8689 :032,136,034,032,136,034,133
8695 :032,136,034,238,138,041,098
```

```
8701 :208,003,238,139,041,173,031
8707 :137,041,208,050,173,149,249
8713 :041,201,003,240,043,201,226
8719 :008,240,039,056,173,138,157
8725 :041,237,140,041,173,139,024
8731 :041,237,141,041,144,024,143
8737 :032,204,255,032,250,019,057
8743 :169,188,160,040,032,108,224
8749 :019,032,125,019,032,223,239
8755 :031,162,001,032,201,255,221
8761 :173,129,041,240,017,141,030
8767 :000,041,169,109,133,253,000
8773 :169,043,133,254,032,101,033
8779 :034,032,103,038,032,136,194
8785 :034,172,134,041,140,145,235
8791 :041,136,136,240,008,048,184
8797 :006,032,136,034,136,208,133
8803 :250,096,169,032,172,131,181
8809 :041,140,144,041,240,006,205
8815 :032,168,031,136,208,250,168
8821 :096,172,136,041,024,152,226
8827 :109,145,041,141,145,041,233
8833 :032,136,034,136,208,250,157
8839 :096,169,013,032,168,031,132
8845 :173,110,045,240,003,032,232
8851 :168,031,096,141,147,041,003
8857 :041,127,032,133,031,174,179
8863 :225,034,221,225,034,240,114
8869 :009,202,208,248,206,144,158
8875 :041,076,242,035,202,138,137
8881 :010,170,140,146,041,169,085
8887 :034,072,169,196,072,189,147
8893 :245,034,072,189,244,034,239
8899 :072,096,056,173,146,041,011
8905 :101,251,133,251,165,252,074
8911 :105,000,133,252,076,238,243
8917 :032,177,251,201,031,240,121
8923 :001,136,140,146,041,096,011
8929 :018,087,065,076,082,084,125
8935 :066,083,078,072,070,064,152
8941 :080,063,088,077,073,071,177
8947 :074,084,035,102,035,111,172
8953 :035,121,035,131,035,141,235
8959 :035,151,035,161,035,176,080
8965 :035,210,035,058,035,074,196
8971 :035,042,035,032,035,023,213
8977 :035,235,035,020,036,093,215
8983 :035,200,169,000,141,143,199
8989 :041,076,214,034,200,032,114
8995 :099,030,141,142,041,076,052
```

```
9001 :214,034,200,032,099,030,138
9007 :141,140,041,173,118,041,189
9013 :141,141,041,076,214,034,188
9019 :200,032,099,030,141,138,187
9025 :041,173,118,041,141,139,206
9031 :041,076,214,034,200,032,156
9037 :099,030,141,133,041,076,085
9043 :214,034,169,000,141,137,010
9049 :041,200,076,214,034,169,055
9055 :010,141,110,045,200,076,165
9061 :214,034,200,169,001,141,092
9067 :150,041,076,214,034,200,054
9073 :032,099,030,141,131,041,075
9079 :076,214,034,200,032,099,006
9085 :030,141,132,041,076,214,247
9091 :034,200,032,099,030,141,155
9097 :134,041,076,214,034,200,068
9103 :032,099,030,141,135,041,109
9109 :076,214,034,200,032,099,036
9115 :030,141,136,041,076,214,025
9121 :034,172,146,041,200,152,138
9127 :072,032,199,033,104,168,007
9133 :140,146,041,096,032,203,063
9139 :035,136,140,129,041,160,052
9145 :001,177,251,153,108,043,150
9151 :200,204,129,041,144,245,130
9157 :240,243,200,076,214,034,180
9163 :200,177,251,201,031,208,247
9169 :249,096,032,203,035,136,192
9175 :140,130,041,160,001,177,096
9181 :251,153,109,044,200,204,158
9187 :130,041,144,245,240,243,246
9193 :076,214,034,032,203,035,059
9199 :076,214,034,200,177,251,167
9205 :201,061,240,007,136,173,039
9211 :147,041,076,009,033,200,245
9217 :032,099,030,072,173,147,042
9223 :041,041,127,170,104,157,135
9229 :237,041,032,214,034,076,135
9235 :197,034,200,162,008,177,029
9241 :251,041,063,201,004,240,057
9247 :009,162,001,201,020,240,152
9253 :003,076,233,031,142,147,157
9259 :028,200,177,251,201,058,190
9265 :240,003,076,233,031,200,064
9271 :177,251,201,031,240,009,196
9277 :032,133,031,153,085,041,024
9283 :076,054,036,152,056,233,162
9289 :003,162,088,160,041,032,047
```

```
9295 :189,255,032,204,255,169,159
9301 :002,032,195,255,169,002,228
9307 :174,147,028,160,000,032,120
9313 :186,255,032,050,019,169,040
9319 :000,166,057,164,058,032,068
9325 :213,255,144,003,076,233,009
9331 :031,142,002,041,140,003,218
9337 :041,104,104,162,001,032,053
9343 :201,255,076,228,032,032,183
9349 :231,255,169,000,032,189,241
9355 :255,169,015,162,008,160,140
9361 :015,032,186,255,032,192,089
9367 :255,144,011,169,015,032,009
9373 :195,255,032,231,255,076,177
9379 :234,019,032,250,019,169,118
9385 :055,160,040,032,108,019,071
9391 :032,199,027,240,022,162,089
9397 :015,032,201,255,176,223,059
9403 :169,048,160,041,032,108,233
9409 :019,169,013,032,210,255,123
9415 :032,204,255,032,231,255,184
9421 :169,000,032,189,255,169,251
9427 :015,162,008,160,015,032,091
9433 :186,255,032,192,255,176,033
9439 :186,032,250,019,162,015,119
9445 :032,198,255,032,199,027,204
9451 :032,204,255,169,015,032,174
9457 :195,255,032,231,255,169,098
9463 :001,141,254,040,096,032,043
9469 :036,037,173,155,041,240,167
9475 :022,032,199,037,032,074,143
9481 :037,173,153,041,201,255,101
9487 :240,009,032,234,037,032,087
9493 :158,018,076,007,037,076,137
9499 :234,019,173,141,002,201,029
9505 :005,208,038,032,250,019,073
9511 :169,209,160,040,032,108,245
9517 :019,032,199,027,141,155,106
9523 :041,208,003,076,234,019,120
9529 :160,000,185,048,041,153,132
9535 :156,041,200,204,005,041,198
9541 :208,244,076,234,019,165,247
9547 :057,133,251,165,058,133,104
9553 :252,169,255,141,153,041,068
9559 :160,001,162,000,173,155,226
9565 :041,240,080,189,156,041,072
9571 :032,026,020,209,251,240,109
9577 :002,162,255,200,208,011,175
9583 :230,252,165,252,205,003,194
9589 :041,240,002,176,054,232,094
```

```
9595 :236,155,041,208,224,024,243
9601 :152,101,251,133,059,165,222
9607 :252,105,000,133,060,173,090
9613 :002,041,197,059,173,003,104
9619 :041,229,060,144,024,056,189
9625 :165,059,237,155,041,133,175
9631 :057,141,152,041,165,060,007
9637 :233,000,133,058,141,153,115
9643 :041,032,134,021,096,032,015
9649 :250,019,169,219,160,040,010
9655 :032,108,019,169,001,141,141
9661 :254,040,096,173,141,002,127
9667 :201,005,208,035,032,250,158
9673 :019,169,229,160,040,032,082
9679 :108,019,032,199,027,141,221
9685 :196,041,240,014,160,000,096
9691 :185,048,041,153,197,041,116
9697 :200,204,005,041,208,244,103
9703 :076,234,019,056,165,057,070
9709 :133,158,237,152,041,133,067
9715 :059,165,058,133,159,237,030
9721 :153,041,005,059,208,101,048
9727 :169,255,141,153,041,024,014
9733 :173,155,041,101,057,133,153
9739 :038,169,000,101,058,133,254
9745 :039,056,173,002,041,229,045
9751 :158,133,180,173,003,041,199
9757 :229,159,133,181,032,035,030
9763 :018,056,173,002,041,237,050
9769 :155,041,141,002,041,173,082
9775 :003,041,233,000,141,003,212
9781 :041,173,196,041,240,041,017
9787 :141,148,041,169,000,141,187
9793 :149,041,032,037,026,160,254
9799 :000,185,197,041,032,026,040
9805 :020,145,057,200,204,196,131
9811 :041,208,242,024,165,057,052
9817 :109,196,041,133,057,165,022
9823 :058,105,000,133,058,076,013
9829 :134,021,160,000,204,000,108
9835 :041,240,032,177,253,048,130
9841 :029,032,133,031,032,252,110
9847 :038,032,168,031,173,151,200
9853 :041,240,010,169,008,032,113
9859 :168,031,169,095,032,168,026
9865 :031,200,076,105,038,096,171
9871 :140,146,041,041,127,141,011
9877 :147,041,032,133,031,201,222
9883 :067,208,027,056,173,142,060
9889 :041,237,000,041,074,056,098
```

```
9895 :237,131,041,168,169,032,177
9901 :032,168,031,136,208,250,230
9907 :172,146,041,076,138,038,022
9913 :201,069,208,017,056,173,141
9919 :132,041,237,000,041,056,186
9925 :237,131,041,168,169,032,207
9931 :076,173,038,201,085,208,216
9937 :008,173,151,041,073,001,144
9943 :141,151,041,201,035,208,224
9949 :018,140,146,041,174,138,110
9955 :041,173,139,041,032,205,090
9961 :221,172,146,041,076,138,003
9967 :038,174,147,041,189,237,041
9973 :041,032,168,031,076,138,219
9979 :038,174,150,041,240,026,152
9985 :133,059,041,127,201,065,115
9991 :144,018,201,091,176,014,139
9997 :170,165,059,041,128,073,137
10003 :128,074,074,133,059,138,113
10009 :005,059,096,032,250,019,230
10015 :056,173,245,040,237,002,016
10021 :041,170,173,246,040,237,176
10027 :003,041,032,205,221,169,202
10033 :001,141,254,040,096,008,077
10039 :014,211,080,069,069,068,054
10045 :211,067,082,073,080,084,146
10051 :032,051,046,048,000,013,001
10057 :018,066,089,032,195,072,033
10063 :065,082,076,069,083,032,230
10069 :194,082,065,078,078,079,149
10075 :078,000,194,085,070,070,076
10081 :069,082,032,195,076,069,108
10087 :065,082,069,068,000,194,069
10093 :085,070,070,069,082,032,005
10099 :198,085,076,076,000,196,234
10105 :069,076,069,084,069,032,008
10111 :040,211,044,215,044,208,121
10117 :041,000,058,032,211,085,048
10123 :082,069,063,032,217,047,137
10129 :206,058,000,197,210,193,241
10135 :211,197,032,193,204,204,168
10141 :000,197,082,065,083,069,141
10147 :032,040,211,044,215,044,237
10153 :208,041,013,018,208,082,227
10159 :069,083,083,032,146,210,030
10165 :197,212,213,210,206,018,213
10171 :032,084,079,032,069,088,059
10177 :073,084,000,208,082,069,197
10183 :083,083,032,070,079,082,116
```

```
10189 :077,065,084,032,075,069,095
10195 :089,058,000,211,065,086,208
10201 :069,058,000,212,065,080,189
10207 :069,032,197,210,210,207,124
10213 :210,000,211,084,079,080,125
10219 :080,069,068,000,214,069,223
10225 :082,073,070,089,032,197,016
10231 :082,082,079,082,000,206,010
10237 :079,032,069,082,082,079,164
10243 :082,083,000,147,032,018,109
10249 :212,146,065,080,069,032,101
10255 :079,082,032,018,196,146,056
10261 :073,083,075,063,000,204,007
10267 :079,065,068,058,000,214,255
10273 :069,082,073,070,089,058,218
10279 :000,208,082,069,083,083,052
10285 :032,018,210,197,212,213,159
10291 :210,206,146,000,196,073,114
10297 :083,075,032,067,079,077,214
10303 :077,065,078,068,058,000,153
10309 :036,206,079,032,210,079,199
10315 :079,077,000,206,079,032,036
10321 :084,069,088,084,032,073,255
10327 :078,032,066,085,070,070,232
10333 :069,082,046,000,147,018,199
10339 :211,146,067,082,069,069,231
10345 :078,044,032,018,196,146,107
10351 :073,083,075,044,032,018,180
10357 :208,146,082,073,078,084,020
10363 :069,082,063,000,196,069,090
10369 :086,073,067,069,032,078,022
10375 :085,077,066,069,082,063,065
10381 :000,211,069,067,079,078,133
10387 :068,065,082,089,032,193,164
10393 :068,068,082,069,083,083,094
10399 :032,035,063,000,198,073,048
10405 :076,069,078,065,077,069,087
10411 :058,000,147,208,082,073,227
10417 :078,084,073,078,071,046,095
10423 :046,046,013,013,000,206,251
10429 :069,088,084,032,083,072,105
10435 :069,069,084,044,032,146,127
10441 :210,197,212,213,210,206,169
10447 :018,000,200,085,078,084,160
10453 :032,070,079,082,058,000,022
10459 :206,079,084,032,198,079,129
10465 :085,078,068,000,210,069,223
10471 :080,076,065,067,069,058,134
10477 :000,209,213,201,212,000,048
```

Program 2-6. Commodore 64 *SpeedScript* File Converter

For mistake-proof program entry, be sure to read "The Automatic Proofreader," earlier in this chapter.

```
100 PRINT"{CLR}{RVS}{N}{2 SPACES}SPEEDSCRIPT FILE
    {SPACE}CONVERSION PROGRAM{3 SPACES}"    :rem 25
110 GOSUB4100                               :rem 167
120 INPUT"{DOWN}INPUT FILE NAME";I$         :rem 113
130 IFI$=""THEN120                          :rem 211
140 INPUT"{DOWN}OUTPUT FILE NAME";O$        :rem 218
150 PRINT"{DOWN}{RVS}D{OFF}ISK, {RVS}S{OFF}CREEN,
    {SPACE}{RVS}P{OFF}RINTER, {RVS}O{OFF}THER"
                                            :rem 29
160 GETA$:IFA$=""THEN160                    :rem 81
170 DV=-(A$="T")-3*(A$="S")-4*(A$="P")-8*(A$="D"):
    SA=7                                    :rem 153
180 IFDV=0THENINPUT"DEVICE NUMBER";DV:INPUT"SECOND
    ARY ADDRESS";SA                         :rem 11
190 PRINT"{2 DOWN}WHICH CONVERSION:"        :rem 192
200 PRINT"{DOWN}1) SPEEDSCRIPT TO COMMODORE ASCII"
                                            :rem 197
210 PRINT"{DOWN}2) SPEEDSCRIPT TO TRUE ASCII"
                                            :rem 98
220 PRINT"{DOWN}3) COMMODORE ASCII TO SPEEDSCRIPT"
                                            :rem 201
230 GETP$:IFP$<"1"ORP$>"3"THEN230           :rem 101
240 ADR=828+VAL(P$)*3-3                     :rem 220
250 OPEN15,8,15,"I0":REM REMOVE ,"I0" IF YOU'VE CH
    ANGED THE DRIVE'S SPEED                 :rem 97
260 OPEN1,8,3,I$:INPUT#15,EN,EM$:F$=I$:IFEN=0THEN2
    90                                      :rem 44
270 PRINT"{DOWN}DISK ERROR FOR ";F$:PRINTEM$
                                            :rem 185
280 PRINT"{3 DOWN}RUN{3 UP}":CLOSE1:CLOSE2:CLOSE15
    :END                                    :rem 48
290 IFDV<>8THENOPEN2,DV,SA,O$:GOTO380       :rem 60
300 EX$=",S,W":IFP$="3"THENEX$=",P,W"       :rem 56
310 OPEN2,DV,SA,"0:"+O$+EX$:INPUT#15,EN,EM$:F$=O$
                                            :rem 42
320 IFEN=0THEN380                           :rem 238
330 IFEN<>63THEN270                         :rem 99
340 IFEN=63THENPRINT"{DOWN}";O$;" EXISTS... REPLAC
    E? {RVS}Y{OFF}/{RVS}N{OFF}:"            :rem 26
350 GETA$:IFA$<>"Y"ANDA$<>"N"THEN350        :rem 45
360 IFA$="N"THEN270                         :rem 36
370 PRINT#15,"S0:"+O$:CLOSE2:GOTO310        :rem 100
380 SYS(ADR):IF(PEEK(144)AND191)=0THENPRINT"{DOWN}
    DONE.":GOTO280                          :rem 184
390 PRINT"I/O ERROR DURING CONVERSION.":INPUT#15,E
    N,EM$:IFEN<>0THEN270                    :rem 253
```

```
400 GOTO280                                    :rem 103
410 FORI=828TO1001:READA:POKEI,A:CK=CK+A:NEXT:IFCK
    =21584THENRETURN                           :rem 222
420 PRINT"{RVS}ERROR IN DATA STATEMENTS.":END
                                               :rem 251
430 DATA 076,069,003,076,122,003               :rem 33
440 DATA 076,174,003,032,225,255               :rem 36
450 DATA 240,018,032,216,003,032               :rem 20
460 DATA 095,003,032,183,255,072               :rem 39
470 DATA 032,224,003,104,041,064               :rem 21
480 DATA 240,233,076,204,255,133               :rem 38
490 DATA 251,041,064,010,005,251               :rem 24
500 DATA 041,191,133,251,041,032               :rem 20
510 DATA 073,032,010,005,251,201               :rem 12
520 DATA 095,208,002,169,013,133               :rem 34
530 DATA 251,096,032,225,255,240               :rem 37
540 DATA 221,032,216,003,032,095               :rem 24
550 DATA 003,041,127,201,065,144               :rem 25
560 DATA 018,201,091,176,014,170               :rem 34
570 DATA 165,251,041,128,073,128               :rem 43
580 DATA 074,074,133,251,138,005               :rem 41
590 DATA 251,133,251,032,183,255               :rem 40
600 DATA 072,032,224,003,104,041               :rem 15
610 DATA 064,240,207,076,204,255               :rem 37
620 DATA 032,225,255,240,169,032               :rem 35
630 DATA 216,003,201,013,208,002               :rem 14
640 DATA 169,031,072,041,128,074               :rem 40
650 DATA 133,251,104,041,063,005               :rem 24
660 DATA 251,133,251,032,183,255               :rem 38
670 DATA 072,032,224,003,104,041               :rem 22
680 DATA 064,240,217,076,204,255               :rem 45
690 DATA 162,001,032,198,255,076               :rem 47
700 DATA 207,255,162,002,032,201               :rem 21
710 DATA 255,165,251,076,210,255               :rem 42
```

Chapter 3
SpeedScript Source Code

Commodore 64 Source Code

The source code for *SpeedScript* was originally developed on the PAL assembler (Pro-Line) and—except for the .ASC and .WORD pseudo-op—is compatible with the LADS assembler from *The Second Book of Machine Language* (COMPUTE! Books, 1984). Line numbers are omitted. Most pseudo-ops are in standard MOS 6502 notation: *= updates the program counter (some assemblers use .ORG, .DB, or .DW instead); .BYT or .BYTE assembles a list of numbers; .WOR or .WORD assembles a list of addresses into low byte/high byte format; .ASC is used to assemble an ASCII character string (many assemblers—including LADS—use .BYTE for this also); < extracts the low byte of a 16-bit expression; > extracts the high byte of a 16-bit expression (some assemblers reverse the use of < and >; others use &255 and /256 to achieve the same effect); and = is used to assign an expression to a label (some assemblers use .EQU).

Beginners should make sure they understand *indirect ,y* addressing, as in LDA ($FB),Y or LDA (CURR),Y. This mode is used extensively in *SpeedScript*.

Notice that a small portion of *SpeedScript* is listed in lowercase. This is how it would actually appear on your screen. It doesn't really matter which mode you're in when typing in the rest of *SpeedScript*—just don't SHIFT to get uppercase.

The VIC version of *SpeedScript* was translated from the 64 source code and developed on the 64. There isn't room to include it here, but it is very similar. Address $BDCD on the 64 becomes $DDCD on the VIC. References to location 1 (which maps in and out ROM in the 64) would be omitted for the VIC. The REFRESH routine, TOPCLR, and a few other routines were changed. The WINDCOLR variable was changed to a subroutine, and the HIGHLIGHT and DELITE routines (which turn on or off the raster interrupt that highlights the command line) were removed. But about 95 percent of the source code did not need to be changed. In fact, the translation only took a single day to get running, and about a week to test and debug.

SpeedScript is written in small modules. Some people think that subroutines are useful only when a routine is called more than once. I strongly believe in breaking up a problem into a number of discrete tasks. These tasks can be written as subroutines, then tested individually. Once all the modules are working, just link them together with JSRs and you have a working program.

I've also tried to use meaningful labels, but sometimes one just runs out of imagination. Comments are added below as signposts to guide you through the source code (you needn't type them in—if you do, precede each comment with a semicolon for the sake of your assembler). Modules are also set apart with blank lines. Notice that some modules are used in rather creative ways. For example, word left/word right is used both for moving the cursor and in delimiting a word to be erased in the erase mode. Also, note that memory locations are sometimes used instead of meaningful labels. In order to fit the complete source code into memory at once, I sometimes had to compromise readability for the sake of brevity.

Crucial to the understanding of *SpeedScript* is the REFRESH routine. Study it carefully. REFRESH is the only time *SpeedScript* writes directly to the screen (Kernal ROM routine $FFD2 is used to print on the command line). It automatically takes care of word-wrap and carriage returns, and provides useful pointers so that the CHECK routine can easily scroll the screen. This frees the rest of *SpeedScript* to just move and modify contiguous memory. Carriage returns are not padded out in memory with spaces to fill the rest of a line; the REFRESH routine takes care of this transparently.

Also, for the sake of compact code, Kernal and BASIC routines are used heavily for routines like Save and Load and for printing numbers.

You'll see some references to location 1, used for mapping the ROMs in and out of the address space. *SpeedScript* stores the main text from the end of the program all the way up to the beginning of I/O space ($CF00). One page of memory is used as a boundary between text areas (the text buffer starts at $D000). This may seem superstitious, but it provides for a margin of error. BASIC is mapped back in when *SpeedScript* needs to call $BDCD to print a number, and then mapped back out. The Kernal ROM is left mapped in, since it is constantly called, but it's mapped out when the program needs to

write to or read from the buffer, which is stored beneath the I/O area and the Kernal. Refer to the memory map shown on page 126.

SpeedScript 3.1 Source Code for Commodore 64

SpeedScript starts at BASIC's normal LOAD address, $0801. These lines simulate the BASIC line **10 SYS 2061** so that *SpeedScript* can be run like any BASIC program.

```
*=    2049
.BYT  $0B,$08,$0A,$00,158
.ASC  "2061"
.BYT  0,0,0
```

Locations used by high-speed memory move routines:

```
FROML   =  $26
FROMH   =  $27
DESTL   =  $9E
DESTH   =  $9F
LLEN    =  $B4
HLEN    =  $B5
```

CURR: Position of cursor within text memory. SCR: used by the REFRESH routine.

```
CURR    =  $39
SCR     =  $C3
```

TEX: An alternate location used in tandem with CURR. COLR is used by REFRESH. TEMP is used throughout as a reusable scratchpad pointer. INDIR is also a reusable indirect pointer. UNDERCURS stores the value of the character highlighted by the cursor.

```
TEX       =  $FB
COLR      =  $14
TEMP      =  $3B
INDIR     =  $FD
UNDERCURS =  $02
```

WINDCOLR: Color of command line window supported by REFRESH. MAP is the 6510's built-in I/O port, used for mapping in and out ROMs from the address space. RETCHAR is the screen-code value of the return mark (a left-pointing arrow).

```
WINDCOLR  =  $0C
MAP       =  $01
RETCHAR   =  31
```

Kernal Routines (refer to the *Commodore 64 Programmer's Reference Guide*):

```
CHROUT  =  $FFD2
STOP    =  $FFE1
SETLFS  =  $FFBA
SETNAM  =  $FFBD
CLALL   =  $FFE7
OPEN    =  $FFC0
CHRIN   =  $FFCF
CHKIN   =  $FFC6
CHKOUT  =  $FFC9
GETIN   =  $FFE4
CLRCHN  =  $FFCC
CLOSE   =  $FFC3
LOAD    =  $FFD5
SAVE    =  $FFD8
IOINIT  =  $FF84
```

Called only when run from BASIC. It is assumed that the author's initials (that conveniently work out in hex) are not normally present in memory. If they are, we know that *SpeedScript* has been run before, so we avoid the ERASE routine to preserve the text in memory.

```
BEGIN     JSR  INIT
          LDA  #$CB
          CMP  FIRSTRUN
          STA  FIRSTRUN
          BEQ  SKIPERAS
          JSR  ERASE
SKIPERAS  JSR  INIT2
          JMP  MAIN
```

UMOVE is a high-speed memory move routine. It gets its speed from self-modifying code (the $0000's at MOVLOOP are replaced by actual addresses when UMOVE is called). Some assemblers may assemble this as a zero-page mode, so you may want to change the $0000's to $FFFF's. UMOVE is used to move an overlapping range of memory upward, so it is used to delete. Set FROML/FROMH to point to the source area of memory, DESTL/DESTH to point to the destination, and LLEN/HLEN to hold the length of the area being moved.

```
UMOVE       LDA  FROML
            STA  MOVLOOP+1
            LDA  FROMH
            STA  MOVLOOP+2
            LDA  DESTL
            STA  MOVLOOP+4
            LDA  DESTH
            STA  MOVLOOP+5
            LDX  HLEN
            BEQ  SKIPMOV
MOV1        LDA  #0
MOV2        STA  ENDPOS
            LDY  #0
MOVLOOP     LDA  $0000,Y
            STA  $0000,Y
            INY
            CPY  ENDPOS
            BNE  MOVLOOP
            INC  MOVLOOP+2
            INC  MOVLOOP+5
            CPX  #0
            BEQ  OUT
            DEX
            BNE  MOV1
SKIPMOV     LDA  LLEN
            BNE  MOV2
OUT         RTS
```

DMOVE uses the same variables as UMOVE, but is used to move an overlapping block of memory downward, so it is used to insert. If the block of memory to be moved does not overlap the destination area, then either routine can be used.

```
DMOVE       LDA  HLEN
            TAX
            ORA  LLEN
            BNE  NOTNULL
            RTS
NOTNULL     CLC
            TXA
            ADC  FROMH
            STA  DMOVLOOP+2
            LDA  FROML
            STA  DMOVLOOP+1
            CLC
            TXA
            ADC  DESTH
            STA  DMOVLOOP+5
            LDA  DESTL
            STA  DMOVLOOP+4
            INX
            LDY  LLEN
            BNE  DMOVLOOP
            BEQ  SKIPDMOV
DMOV1       LDY  #255
DMOVLOOP    LDA  $0000,Y
            STA  $0000,Y
            DEY
            CPY  #255
            BNE  DMOVLOOP
SKIPDMOV    DEC  DMOVLOOP+2
            DEC  DMOVLOOP+5
            DEX
            BNE  DMOV1
            RTS
```

REFRESH copies a screenful of text from the area of memory pointed to by TOPLIN. It works like a printer routine, fitting a line of text between the screen margins, wrapping words, and restarts at the left margin after printing a carriage return. *SpeedScript* constantly calls this routine while the cursor is blinking, so it has to be very fast. To eliminate flicker, it also clears out the end of each line instead of first clearing the screen. It stores the length of the first screen line for the sake of the CHECK routine (which scrolls up by adding that length to TOPLIN), the last text location referenced (so CHECK can see if the cursor has moved off the visible screen).

```
REFRESH     LDA  #40
            STA  SCR
            STA  COLR
            LDA  #4
            STA  SCR+1
            LDA  #$D8
            STA  COLR+1
            LDA  TOPLIN
            STA  TEX
            LDA  TOPLIN+1
            STA  TEX+1
            LDX  #1
            LDA  INSMODE
            STA  WINDCOLR
            LDA  SCRCOL
            STA  53280
PPAGE       LDY  #0
PLINE       LDA  TEXCOLR
            STA  (COLR),Y
            LDA  (TEX),Y
            STA  LBUFF,Y
            INY
            AND  #127
            CMP  #RETCHAR
            BEQ  BREAK
            CPY  #40
            BNE  PLINE
            DEY
SLOOP       LDA  (TEX),Y
            AND  #127
NXCUR       CMP  #32
            BEQ  SBRK
            DEY
            BNE  SLOOP
            LDY  #39
SBRK        INY
```

```
BREAK   STY  TEMP
        DEY
COPY    LDA  LBUFF,Y
        STA  (SCR),Y
        DEY
        BPL  COPY
        LDY  TEMP
        CLC
        TYA
        ADC  TEX
        STA  TEX
        LDA  TEX+1
        ADC  #0
        STA  TEX+1
        CPX  #1
        BNE  CLRLN
        STY  LENTABLE
CLRLN   CPY  #40
        BEQ  CLEARED
        LDA  #32
        STA  (SCR),Y
        INY
        JMP  CLRLN
CLEARED CLC
        LDA  SCR
        ADC  #40
        STA  SCR
        STA  COLR
        BCC  INCNOT
        INC  SCR+1
        INC  COLR+1
INCNOT  INX
        CPX  #25
        BEQ  PDONE
        JMP  PPAGE
PDONE   LDA  TEX
        STA  BOTSCR
        LDA  TEX+1
        STA  BOTSCR+1
        RTS
```

The following routine fills the entire text area with space characters (screen code 32), effectively erasing all text. It is called when the program is first run, and when an Erase All is performed.

```
ERASE   LDA  TEXSTART
        STA  TEX
        STA  TOPLIN
        STA  LASTLINE
        STA  CURR
        LDA  TEXSTART+1
        STA  TEX+1
        STA  TOPLIN+1
        STA  LASTLINE+1
        STA  CURR+1
        SEC
        LDA  TEXEND+1
        SBC  TEXSTART+1
        TAX
        LDA  #32
CLRLOOP LDY  #255
        DEC  TEX+1
        STA  (TEX),Y
        INY
        INC  TEX+1
CLR2    STA  (TEX),Y
        INY
        BNE  CLR2
        INC  TEX+1
        DEX
        BNE  CLR2
        STA  (TEX),Y
        RTS
```

PRMSG is used anytime we need to print something at the top of the screen (the command line). Pass it the address of the message to be printed by storing the low byte of the address in the accumulator, and the high byte in the Y register. The message in memory must end with a zero byte. The routine does not add a carriage return.

```
PRMSG   STA  TEMP
        STY  TEMP+1
        LDY  #0
PRLOOP  LDA  (TEMP),Y
        BEQ  PREXIT
        JSR  CHROUT
        INY
        BNE  PRLOOP
PREXIT  RTS
GETAKEY JSR  GETIN
        BEQ  GETAKEY
        RTS
```

The initialization routine sets up the memory map, clears out certain flags, and enables the raster interrupt.

```
INIT    LDA  #147
        JSR  CHROUT
        LDA  #54
        STA  MAP
        LDA  #0
        STA  INSMODE
        STA  TEXSTART
        STA  TEXEND
        STA  TEXBUF
        STA  BUFEND
        STA  HUNTLEN
        STA  REPLEN
        LDA  #>END
        CLC
        ADC  #1
        STA  TEXSTART+1
        LDA  #$CF
        STA  TEXEND+1
        LDA  #$D0
        STA  TEXBUF+1
        LDA  #$FF
        STA  BUFEND+1
        STA  FPOS+1
        JMP  IOINIT
```

```
INIT2     JSR   KILLBUFF
          LDA   #128
          STA   650
          STA   $9D
          JSR   HIGHLIGHT
          LDA   #<MYNMI
          STA   $318
          LDA   #>MYNMI
          STA   $319
          LDA   TEXSTART
          STA   CURR
          LDA   TEXSTART+1
          STA   CURR+1
          JSR   SYSMSG
          LDA   #<MSG2
          LDY   #>MSG2
          JSR   PRMSG
          INC   MSGFLG
          RTS
```

The NOPS are here because I replaced a three-byte JSR CHECK with RTS. I did not want the size of the code or the location of any routines to change. JSR CHECK was originally inserted to fix a bug, but caused a bug itself.

```
          NOP
          NOP
```

SYSMSG displays "SpeedScript 3.1". The message flag (MSGFLG) is set when a message is to be left on the screen only until the next keystroke. After that keystroke, SYSMSG is called. The INIT routine also prints the credit line with the MSGFLG set so that you won't have to stare at the author's name while you're writing—a modesty feature.

```
SYSMSG    JSR   TOPCLR
          LDA   #<MSG1
          LDY   #>MSG1
          JSR   PRMSG
          LDA   #0
          STA   MSGFLG
          RTS
```

This routine traps the RESTORE key. It reproduces some of the ROM code so that RS-232 is still supported (although *SpeedScript* does not directly support RS-232 output).

```
MYNMI     PHA
          TXA
          PHA
          TYA
          PHA
          LDA   #$7F
          STA   $DD0D
          LDY   $DD0D
          BPL   NOTRS
          JMP   $FE72
```

If RESTORE is pressed, we have to fix the cursor in case it was lit.

```
NOTRS     LDA   BLINKFLAG
          BEQ   NOTCURSOR
          LDA   UNDERCURS
          LDY   #0
          STA   (CURR),Y
NOTCURSOR LDA   #2
          STA   WINDCOLR
          JSR   CLRCHN
          JSR   TOPCLR
          LDA   #<XITMSG
          LDY   #>XITMSG
          JSR   PRMSG
          JSR   YORN
          BNE   REBOOT
          JSR   DELITE
          SEI
          LDA   #$7F
          JMP   $FE66
REBOOT    JSR   DELITE
          LDX   #$FA
          TXS
          JSR   INIT2
          JMP   MAIN
```

TOPCLR keeps the command line clean. It is called before most messages. It's like a one-line clear-screen.

```
TOPCLR    LDX   #39
          LDA   #32
TOPLOOP   STA   1024,X
          DEX
          BPL   TOPLOOP
          LDA   #19
          JMP   CHROUT
```

Converts Commodore ASCII to screen codes.

```
ASTOIN    PHA
          AND   #128
          LSR
          STA   TEMP
          PLA
          AND   #63
          ORA   TEMP
          RTS
```

The MAIN loop blinks the cursor, checks for keystrokes, converts them from ASCII to screen codes, puts them in text at the CURRent position, and increments the CURRent position and LASTLINE. It also checks for special cases like the back arrow and the return key, and passes control characters to the CONTROL routine. SHIFTed spaces are turned into unSHIFTed ones. The INSMODE flag is checked to see if we should insert a space before a character.

```
MAIN        LDY  #0
            STY  BLINKFLAG
            LDA  (CURR),Y
            STA  UNDERCURS
MAIN2       LDY  #0
            LDA  (CURR),Y
            EOR  #$80
            STA  (CURR),Y
            LDA  BLINKFLAG
            EOR  #1
            STA  BLINKFLAG
            JSR  REFRESH
WAIT        JSR  GETIN
            BNE  KEYPRESS
            LDA  162
            AND  #16
            BEQ  WAIT
            LDA  #0
            STA  162
            JMP  MAIN2
KEYPRESS    TAX
            LDY  #0
            LDA  UNDERCURS
            STA  (CURR),Y
            STY  BLINKFLAG
            CPX  #95
            BNE  NOTBKS
            JSR  LEFT
            LDA  #32
            LDY  #0
            STA  (CURR),Y
            JMP  MAIN
NOTBKS      LDA  MSGFLG
            BEQ  NOMSG
            TXA
            PHA
            JSR  SYSMSG
            PLA
            TAX
NOMSG       TXA
            CMP  #13
            BNE  NOTCR
            LDX  #RETCHAR+64
NOTCR       TXA
            AND  #127
            CMP  #32
            BCC  CONTROL
            CPX  #160
            BNE  NESHIFT
            LDX  #32
NESHIFT     TXA
            PHA
            LDY  #0
            LDA  (CURR),Y
            CMP  #RETCHAR
            BEQ  DOINS
            LDA  INSMODE
            BEQ  NOTINST
DOINS       JSR  INSCHAR
NOTINST     PLA
            JSR  ASTOIN
PUTCHR      LDY  #0
            STA  (CURR),Y
            JSR  REFRESH
            SEC
            LDA  CURR
            SBC  LASTLINE
            STA  TEMP
            LDA  CURR+1
            SBC  LASTLINE+1
            ORA  TEMP
            BCC  INKURR
            LDA  CURR
            ADC  #0
            STA  LASTLINE
            LDA  CURR+1
            ADC  #0
            STA  LASTLINE+1
INKURR      INC  CURR
            BNE  NOINC2
            INC  CURR+1
NOINC2      JSR  CHECK
            JMP  MAIN
```

CONTROL looks up a keyboard command in the list of control codes at CTBL. The first byte of CTBL is the actual number of commands. Once the position is found, this position is doubled as an index to the two-byte address table at VECT. The address of MAIN−1 is put on the stack, simulating the return address; then the address of the command routine taken from VECT is pushed. We then perform an RTS. RTS pulls the bytes off the stack as if they were put there by a JSR. This powerful technique is used to simulate ON-GOTO in machine language.

```
CONTROL     TXA
            LDX  CTBL
SRCH        CMP  CTBL,X
            BEQ  FOUND
            DEX
            BNE  SRCH
            JMP  MAIN
FOUND       DEX
            TXA
            ASL  A
            TAX
            LDA  #>MAIN-1
            PHA
            LDA  #<MAIN-1
            PHA
            LDA  VECT+1,X
            PHA
            LDA  VECT,X
            PHA
            RTS
CTBL        .BYT 39
            .BYT 29,157,137,133,2,12,
                 138,134,20,148
            .BYT 4,19,9,147,135,139,5
                 ,136,140
```

```
        .BYT  22,145,17,159,18,24,
              26,16
        .BYT  28,30,6,1,11,8,31,3,1
              31
        .BYT  10,141,7
VECT    .WOR  RIGHT-1,LEFT
              -1,WLEFT-1,W
              RIGHT-1,BORD
              ER-1,LETTERS
              -1
        .WOR  SLEFT-1,SRIGH
              T-1,DELCHAR
              -1,INSCHAR-1,
              DELETE-1
        .WOR  HOME-1,INSTG
              L-1,CLEAR-1,P
              ARIGHT-1,PAR
              LEFT-1
        .WOR  ERAS-1,TLOAD
              -1,TSAVE-1,VE
              RIFY-1
        .WOR  SLEFT-1,SRIGH
              T-1,CATALOG
              -1,INSBUFFER-
              1,SWITCH-1
        .WOR  ENDTEX-1,PRI
              NT-1,FORMAT
              -1,DCMND-1
        .WOR  DELIN-1,ALPH
              A-1,KILLBUFF-
              1,HUNT-1,FREE
              MEM-1,TAB-1
        .WOR  LOTTASPACES
              -1,REPSTART-
              1,ENDPAR-1,SA
              NDR-1
```

The check routine first prevents the cursor from disappearing past the beginning or end-of-text memory, and prevents us from cursoring past the end-of-text pointer. It also checks to see if the cursor has left the visible screen, scrolling with REFRESH to make the cursor visible. The double-byte SBCs are used as a 16-bit CMP macro, setting the Z and C flags just like CMP does.

```
CHECK   JSR   CHECK2
        SEC
        LDA   CURR
        SBC   TOPLIN
        LDA   CURR+1
        SBC   TOPLIN+1
        BCS   OK1
        SEC
        LDA   TOPLIN
        SBC   TEXSTART
        STA   TEMP
        LDA   TOPLIN+1
        SBC   TEXSTART+1
        ORA   TEMP
        BEQ   OK1
        LDA   CURR
        STA   TOPLIN
        LDA   CURR+1
        STA   TOPLIN+1
        JSR   REFRESH
OK1     SEC
        LDA   BOTSCR
        SBC   CURR
        STA   TEX
        LDA   BOTSCR+1
        SBC   CURR+1
        STA   TEX+1
        ORA   TEX
        BEQ   EQA
        BCS   OK2
EQA     CLC
        LDA   TOPLIN
        ADC   LENTABLE
        STA   TOPLIN
        LDA   TOPLIN+1
        ADC   #0
        STA   TOPLIN+1
REF     JSR   REFRESH
        JMP   OK1
OK2     RTS
CHECK2  SEC
        LDA   LASTLINE
        SBC   TEXEND
        STA   TEMP
        LDA   LASTLINE+1
        SBC   TEXEND+1
        ORA   TEMP
        BCC   CK3
        LDA   TEXEND
        STA   LASTLINE
        LDA   TEXEND+1
        STA   LASTLINE+1
CK3     SEC
        LDA   CURR
        SBC   TEXSTART
        STA   TEMP
        LDA   CURR+1
        SBC   TEXSTART+1
        ORA   TEMP
        BCS   INRANGE
        LDA   TEXSTART
        STA   CURR
        LDA   TEXSTART+1
        STA   CURR+1
        RTS
INRANGE SEC
        LDA   CURR
        SBC   LASTLINE
        STA   TEMP
        LDA   CURR+1
        SBC   LASTLINE+1
        ORA   TEMP
        BCS   OUTRANGE
        RTS
OUTRANGE LDA  LASTLINE
        STA   CURR
```

```
            LDA   LASTLINE+1
            STA   CURR+1
            RTS
```

Move cursor right.

```
RIGHT       INC   CURR
            BNE   NOINCR
            INC   CURR+1
NOINCR      JMP   CHECK
```

Cursor left.

```
LEFT        LDA   CURR
            BNE   NODEC
            DEC   CURR+1
NODEC       DEC   CURR
            JMP   CHECK
```

Word left. We look backward for a space.

```
WLEFT       LDA   CURR
            STA   TEX
            LDA   CURR+1
            STA   TEX+1
            DEC   TEX+1
            LDY   #$FF
STRIP       LDA   (TEX),Y
            CMP   #32
            BEQ   STRLOOP
            CMP   #RETCHAR
            BNE   WLOOP
STRLOOP     DEY
            BNE   STRIP
WLOOP       LDA   (TEX),Y
            CMP   #32
            BEQ   WROUT
            CMP   #RETCHAR
            BEQ   WROUT
            DEY
            BNE   WLOOP
            RTS
WROUT       SEC
            TYA
            ADC   TEX
            STA   CURR
            LDA   TEX+1
            ADC   #0
            STA   CURR+1
            JMP   CHECK
```

Word right. We scan forward for a space. OIDS is not a meaningful label.

```
WRIGHT      LDY   #0
RLOOP       LDA   (CURR),Y
            CMP   #32
            BEQ   ROUT
            CMP   #RETCHAR
            BEQ   ROUT
            INY
            BNE   RLOOP
            RTS
ROUT        INY
            BNE   OIDS
            INC   CURR+1
            LDA   CURR+1
            CMP   LASTLINE+1
            BCC   OIDS
            BNE   LASTWORD
OIDS        LDA   (CURR),Y
            CMP   #32
            BEQ   ROUT
            CMP   #RETCHAR
            BEQ   ROUT
```

Add the Y register to the CURRent cursor position to move the cursor. CHECK prevents illegal cursor movement. LASTWORD is called if the end of the word cannot be found with 255 characters.

```
ADYCURR     CLC
            TYA
            ADC   CURR
            STA   CURR
            LDA   CURR+1
            ADC   #0
            STA   CURR+1
WRTN        JMP   CHECK

LASTWORD    LDA   LASTLINE
            STA   CURR
            LDA   LASTLINE+1
            STA   CURR+1
            JMP   CHECK
```

ENDTEX is tricky. If the end-of-text pointer would point to an area already visible on the screen, we just move the cursor there and call REFRESH. Otherwise, we step back 1K from the end-of-text and then scroll to the end. This is necessary since in worst case only 24 characters of return marks would fill the screen.

```
ENDTEX      LDA   #0
            STA   TOPLIN
            LDA   LASTLINE+1
            SEC
            SBC   #4
            CMP   TEXSTART+1
            BCS   SAFE
            LDA   TEXSTART+1
SAFE        STA   TOPLIN+1
            JSR   REFRESH
            JMP   LASTWORD
```

The raster interrupt automatically places SCRCOL into 53281 when appropriate. The AND keeps SCRCOL within a legal range (I know that's not really necessary).

```
BORDER      INC   SCRCOL
            LDA   SCRCOL
            AND   #15
```

```
                STA   SCRCOL
                RTS
SCRCOL          .BYTE 12
```

TEXCOLR (text color) is used in the REFRESH routine and stored into color memory. Both SCRCOL and TEXCOLR are stored within the *SpeedScript* code so that after they're changed, you can resave *SpeedScript* and it will come up with your color choice in the future.

```
LETTERS         INC   TEXCOLR
                LDA   TEXCOLR
                AND   #15
                STA   TEXCOLR
                JMP   REFRESH
TEXCOLR         .BYTE 11
```

Sentence left. We look backward for ending punctuation or a return mark, then go forward until we run out of spaces.

```
SLEFT           LDA   CURR
                STA   TEX
                LDA   CURR+1
                STA   TEX+1
                DEC   TEX+1
                LDY   #$FF
PMANY           LDA   (TEX),Y
                CMP   #"."
                BEQ   PSRCH
                CMP   #"!"
                BEQ   PSRCH
                CMP   #"?"
                BEQ   PSRCH
                CMP   #RETCHAR
                BNE   PSLOOP
PSRCH           DEY
                BNE   PMANY
                RTS
PSLOOP          LDA   (TEX),Y
                CMP   #"."
                BEQ   PUNCT
                CMP   #"!"
                BEQ   PUNCT
                CMP   #"?"
                BEQ   PUNCT
                CMP   #RETCHAR
                BEQ   PUNCT
                DEY
                BNE   PSLOOP
                DEC   TEX+1
                LDA   TEX+1
                CMP   TEXSTART
                BCS   PSLOOP
                JMP   FIRSTWORD
PUNCT           STY   TEMP
                DEC   TEMP
SKIPSPC         INY
                BEQ   REPEAT
                LDA   (TEX),Y
                CMP   #32
                BEQ   SKIPSPC
                DEY
                JMP   WROUT
REPEAT          LDY   TEMP
                JMP   PSLOOP
FIRSTWORD       LDA   TEXSTART
                STA   CURR
                LDA   TEXSTART+1
                STA   CURR+1
                JMP   CHECK
```

Sentence right. We look forward for ending punctuation, then skip forward until we run out of spaces.

```
SRIGHT          LDY   #0
SRLP            LDA   (CURR),Y
                CMP   #"."
                BEQ   PUNCT2
                CMP   #"!"
                BEQ   PUNCT2
                CMP   #"?"
                BEQ   PUNCT2
                CMP   #RETCHAR
                BEQ   PUNCT2
                INY
                BNE   SRLP
                INC   CURR+1
                LDA   CURR+1
                CMP   LASTLINE+1
                BEQ   SRLP
                BCC   SRLP
SREXIT          JMP   LASTWORD
PUNCT2          INY
                BNE   NOFIXCURR
                INC   CURR+1
                LDA   CURR+1
                CMP   LASTLINE+1
                BCC   NOFIXCURR
                BEQ   NOFIXCURR
                JMP   LASTWORD
NOFIXCURR       LDA   (CURR),Y
                CMP   #32
                BEQ   PUNCT2
                CMP   #"."
                BEQ   PUNCT2
                CMP   #"!"
                BEQ   PUNCT2
                CMP   #"?"
                BEQ   PUNCT2
                CMP   #RETCHAR
                BEQ   PUNCT2
                JMP   ADYCURR
```

The text buffer starts at a fixed location, but the end of the buffer is changed as text is added to it. To clear the buffer, we just set the end of the buffer to the value of the start of the buffer. No text is actually erased.

```
KILLBUFF        LDA   TEXBUF
                STA   TPTR
                LDA   TEXBUF+1
```

```
        STA   TPTR+1
        JSR   TOPCLR
        LDA   #<KILLMSG
        LDY   #>KILLMSG
        JSR   PRMSG
        LDA   #1
        STA   MSGFLG
        RTS
```

This is the second level of the general-purpose delete routines. UMOVE is the primitive core of deleting. For CTRL-D, the current cursor position is the source, then a cursor command is called to update the cursor pointer. This becomes the destination. For CTRL-E, the current cursor position is the destination, a cursor routine is called, and this becomes the source. UMOVE is then called. We actually move more than the length from the source to the end-of-text. Some extra text is moved from past the end-of-text. Since everything past the end-of-text is spaces, this neatly erases everything past the new end-of-text position. Naturally, the end-of-text pointer is updated. Before the actual delete is performed, the text to be deleted is stored in the buffer so that it can be recalled in case of error. The buffer doubles as a fail-safe device and for moving and copying text. Checks are made to make sure that the buffer does not overflow.

```
DEL1    SEC
        LDA   CURR
        SBC   TEXSTART
        STA   TEMP
        LDA   CURR+1
        SBC   TEXSTART+1
        ORA   TEMP
        BNE   DEL1A
DELABORT PLA
        PLA
        RTS

DEL1A   LDA   CURR
        STA   FROML
        LDA   CURR+1
        STA   FROMH
        RTS
DEL2    SEC
        LDA   CURR
        STA   DESTL
        EOR   #$FF
        ADC   FROML
        STA   GOBLEN
        LDA   CURR+1
        STA   DESTH
        EOR   #$FF
        ADC   FROMH
        STA   GOBLEN+1

DELC    LDA   FROML
        STA   FROMSAV
        LDA   FROMH
        STA   FROMSAV+1
        LDA   DESTL
        STA   DESTSAV
        STA   FROML
        LDA   DESTH
        STA   DESTSAV+1
        STA   FROMH
        SEC
        LDA   GOBLEN+1
        ADC   TPTR+1
        CMP   BUFEND+1
        BCC   GOSAV
        JSR   TOPCLR
        LDA   #<BUFERR
        LDY   #>BUFERR
        JSR   PRMSG
        LDA   #1
        STA   MSGFLG
        LDA   #0
        STA   198
        RTS

GOSAV   LDA   TPTR
        STA   DESTL
        LDA   TPTR+1
        STA   DESTH
        LDA   GOBLEN
        STA   LLEN
        CLC
        ADC   TPTR
        STA   TPTR
        LDA   GOBLEN+1
        STA   HLEN
        ADC   TPTR+1
        STA   TPTR+1
        LDA   #0
        STA   $D01A
        LDA   #52
        STA   MAP
        JSR   UMOVE
        LDA   #54
        STA   MAP
        LDA   #1
        STA   $D01A

        LDA   FROMSAV
        STA   FROML
        LDA   FROMSAV+1
        STA   FROMH
        LDA   DESTSAV
        STA   DESTL
        LDA   DESTSAV+1
        STA   DESTH
        SEC
        LDA   LASTLINE
        SBC   DESTL
        STA   LLEN
        LDA   LASTLINE+1
```

```
        SBC   DESTH
        STA   HLEN
        JSR   UMOVE
        SEC
        LDA   LASTLINE
        SBC   GOBLEN
        STA   LASTLINE
        LDA   LASTLINE+1
        SBC   GOBLEN+1
        STA   LASTLINE+1
        RTS
```

Most delete commands end up calling the above routines. The single-character deletes must subtract 1 from the buffer pointer so that single characters are not added to the buffer. But note how short these routines are.

```
DELCHAR JSR   DEL1
        JSR   LEFT
        JSR   DEL2

FIXTP   SEC
        LDA   TPTR
        SBC   #1
        STA   TPTR
        LDA   TPTR+1
        SBC   #0
        STA   TPTR+1
        RTS
```

This is called from CTRL–back arrow. We first check to see if SHIFT is also held down. If so, we go to another routine that "eats" spaces.

```
DELIN   LDA   653
        CMP   #5
        BNE   DODELIN
        JMP   EATSPACE
DODELIN JSR   RIGHT
        JSR   DEL1
        JSR   LEFT
        JSR   DEL2
        JMP   FIXTP
```

Called by CTRL-D. As mentioned, it stores CURR into FROML/FROMH, moves the cursor either by sentence, word, or paragraph, then stores the new position of CURR into DESTL and DESTH. The above routines perform the actual delete. CTRL-D always discards the previous contents of the buffer, for reasons that are obvious once you think about what would happen to the buffer if we didn't clear it. Notice how we change the color of the command window to red (color 2) to warn the user of the impending deletion.

```
DELETE  JSR   KILLBUFF
        LDA   #2
        STA   WINDCOLR
        JSR   TOPCLR
        LDA   #<DELMSG
        LDY   #>DELMSG
        JSR   PRMSG
        JSR   GETAKEY
        PHA
        JSR   SYSMSG
        PLA
        AND   #191
        CMP   #23
        BNE   NOTWORD

DELWORD JSR   DEL1
        JSR   WLEFT
        JMP   DEL2
NOTWORD CMP   #19
        BNE   NOTSENT

DELSENT JSR   DEL1
        JSR   SLEFT
        JMP   DEL2
NOTSENT CMP   #16
        BNE   NOTPAR
        JSR   DEL1
        JSR   PARLEFT
        JMP   DEL2
NOTPAR  RTS
```

Home the cursor. If the cursor is already home, move the cursor to the top of text.

```
HOME    SEC
        LDA   CURR
        SBC   TOPLIN
        STA   TEMP
        LDA   CURR+1
        SBC   TOPLIN+1
        ORA   TEMP
        BEQ   TOPHOME
        LDA   TOPLIN
        STA   CURR
        LDA   TOPLIN+1
        STA   CURR+1
        RTS

TOPHOME LDA   TEXSTART
        STA   CURR
        LDA   TEXSTART+1
        STA   CURR+1
        JMP   CHECK
```

This deletes all spaces between the cursor and following nonspace text. Sometimes inventing labels can be fun.

```
EATSPACE LDA  CURR
         STA  TEX
         STA  DESTL
         LDA  CURR+1
         STA  TEX+1
         STA  DESTH
```

```
            LDY   #0
SPCSRCH     LDA   (TEX),Y
            CMP   #32
            BNE   OUTSPACE
            INY
            BNE   SPCSRCH
            LDA   TEX+1
            CMP   LASTLINE+1
            BCC   GOINC
            LDA   LASTLINE
            STA   TEX
            LDA   LASTLINE+1
            STA   TEX+1
            LDY   #0
            JMP   OUTSPACE
GOINC       INC   TEX+1
            JMP   SPCSRCH
OUTSPACE    CLC
            TYA
            ADC   TEX
            STA   FROML
            LDA   #0
            ADC   TEX+1
            STA   FROMH
            SEC
            LDA   LASTLINE
            SBC   DESTL
            STA   LLEN
            LDA   LASTLINE+1
            SBC   DESTH
            STA   HLEN
            SEC
            LDA   FROML
            SBC   DESTL
            STA   GOBLEN
            LDA   FROMH
            SBC   DESTH
            STA   GOBLEN+1
            JSR   UMOVE
            SEC
            LDA   LASTLINE
            SBC   GOBLEN
            STA   LASTLINE
            LDA   LASTLINE+1
            SBC   GOBLEN+1
            STA   LASTLINE+1
            RTS
```

Inserts 255 spaces. Notice how it and other insert routines use TAB2.

```
LOTTASPACES LDA   #255
            STA   INSLEN
            JMP   TAB2
TAB         LDA   #5
            STA   INSLEN
            JSR   TAB2
            LDA   (CURR),Y
            CMP   #32
            BNE   NOINCY
            INY
NOINCY      JMP   ADYCURR
TAB2        LDA   #0
            STA   INSLEN+1
            JSR   INSBLOCK
            LDA   #32
            LDX   INSLEN
            LDY   #0
FILLSP      STA   (CURR),Y
            INY
            DEX
            BNE   FILLSP
            RTS
```

SHIFT–RETURN calls this. It inserts two spaces, fills them with return marks, then calls TAB for a margin indent. Not much code for a useful routine.

```
ENDPAR      JSR   INSCHAR
            JSR   INSCHAR
            LDA   #RETCHAR
            LDY   #0
            STA   (CURR),Y
            INY
            STA   (CURR),Y
            JSR   REFRESH
            JSR   RIGHT
            JSR   RIGHT
            JMP   TAB
```

Insert a single space.

```
INSCHAR     LDA   #1
            STA   INSLEN
            LDA   #0
            STA   INSLEN+1
            JSR   INSBLOCK
            LDA   #32
            LDY   #0
            STA   (CURR),Y
            JMP   CHECK
```

A general routine to insert as many spaces as are specified by INSLEN.

```
INSBLOCK    CLC
            LDA   LASTLINE
            ADC   INSLEN
            LDA   LASTLINE+1
            ADC   INSLEN+1
            CMP   TEXEND+1
            BCC   OKINS
            PLA
            PLA
            JMP   INOUT
OKINS       CLC
            LDA   CURR
            STA   FROML
            ADC   INSLEN
            STA   DESTL
            LDA   CURR+1
            STA   FROMH
            ADC   INSLEN+1
            STA   DESTH
            SEC
            LDA   LASTLINE
            SBC   FROML
```

```
            STA   LLEN
            LDA   LASTLINE+1
            SBC   FROMH
            STA   HLEN
            JSR   DMOVE
            CLC
            LDA   LASTLINE
            ADC   INSLEN
            STA   LASTLINE
            LDA   LASTLINE+1
            ADC   INSLEN+1
            STA   LASTLINE+1
INOUT       RTS
```

Toggle insert mode. The INSMODE flag doubles as the color of the command line.

```
INSTGL      LDA   INSMODE
            EOR   #14
            STA   INSMODE
            RTS
```

Another example of modular code. This is called anytime a yes/no response is called for. It prints "Are you sure? (Y/N)", then returns with the zero flag set to true if Y was pressed, ready for the calling routine to use BEQ or BNE as a branch for yes or no.

```
YORN        LDA   #<YMSG
            LDY   #>YMSG
            JSR   PRMSG
YORNKEY     JSR   $FF9F
            JSR   GETIN
            BEQ   YORNKEY
            CMP   #147
            BEQ   YORNKEY
            AND   #127
            CMP   #"Y"
            RTS
```

Erase all text. Calls YORN to affirm the deadly deed, then calls ERASE to erase all text, INIT2 to reset some flags, then jumps back to the main loop. *LDX #$FA:TXS* is used to clean up the stack. If you would prefer to have the buffer contents preserved after an Erase All, change the *JSR INIT2* in the following routine to *JSR INIT2+3*.

```
CLEAR       LDA   #2
            STA   WINDCOLR
            JSR   TOPCLR
            LDA   #<CLRMSG
            LDY   #>CLRMSG
            JSR   PRMSG
            JSR   YORN
            BEQ   DOIT
            JMP   SYSMSG
DOIT        LDX   #$FA
            TXS
            JSR   ERASE
            JSR   INIT2
            JMP   MAIN
```

Paragraph right. What's this routine doing here instead of with the other cursor routines? You don't always write your routines in the order of a flowchart. I didn't originally plan to have a paragraph movement function, so I added it where there was room for it between line numbers.

```
PARIGHT     LDY   #0
PARLP       LDA   (CURR),Y
            CMP   #RETCHAR
            BEQ   RETFOUND
            INY
            BNE   PARLP
            INC   CURR+1
            LDA   CURR+1
            CMP   LASTLINE+1
            BCC   PARLP
            BEQ   PARLP
            JMP   LASTWORD
RETFOUND    INY
            BNE   GOADY
            INC   CURR+1
GOADY       JMP   ADYCURR
```

Paragraph left. Notice the trick of decrementing the high byte of the pointer, then starting the index at 255 in order to search backward.

```
PARLEFT     LDA   CURR
            STA   TEX
            LDA   CURR+1
            STA   TEX+1
            DEC   TEX+1
            LDY   #$FF
PARLOOP     LDA   (TEX),Y
            CMP   #RETCHAR
            BEQ   RETF2
PARCONT     DEY
            CPY   #255
            BNE   PARLOOP
            DEC   TEX+1
            LDA   TEX+1
            CMP   TEXSTART+1
            BCS   PARLOOP
            JMP   FIRSTWORD
RETF2       SEC
            TYA
            ADC   TEX
            STA   TEX
            LDA   #0
            ADC   TEX+1
            STA   TEX+1
            SEC
            LDA   TEX
            SBC   CURR
            STA   TEMP
            LDA   TEX+1
```

```
            SBC   CURR+1
            ORA   TEMP
            BNE   TEXTOCURR
            STY   TEMP
            CLC
            LDA   TEX
            SBC   TEMP
            STA   TEX
            LDA   TEX+1
            SBC   #0
            STA   TEX+1
            JMP   PARCONT
TEXTOCURR   LDA   TEX
            STA   CURR
            LDA   TEX+1
            STA   CURR+1
            JMP   CHECK
```

This enables the raster interrupt. The raster interrupt allows separate background colors for the command line and the rest of the screen. It lets us change the color of the top line to flag insert mode or to warn the user with a red color that he/she should be careful. Since it is an interrupt, it is always running in the background. Interrupt routines must always be careful not to corrupt the main program.

```
HIGHLIGHT   SEI
            LDA   #0
            STA   $DC0E
            LDA   #27
            STA   $D011
            LDA   #<IRQ
            STA   $314
            LDA   #>IRQ
            STA   $315
            LDA   #1
            STA   $D01A
            STA   $D012
            CLI
            RTS
IRQ         LDA   #58
            LDY   WINDCOLR
            CMP   $D012
            BNE   MID
            LDA   #1
            LDY   SCRCOL
MID         STY   $D021
            STA   $D012
SKIP        CMP   #1
            BEQ   DEFALT
            LDA   #1
            STA   $D019
            JMP   $FEBC
DEFALT      LDA   #1
            STA   $D019
            JMP   $EA31
```

ERAS is called by CTRL-E. It works much like CTRL-D. Notice that the ORA #64 allows users to press either S, W, P, or CTRL-S, CTRL-W, CTRL-P, in case they have a habit of leaving the control key held down. It must call REFRESH after each move and adjust the new position of the cursor. If SHIFT is held down with CTRL-E, we don't erase the previous contents of the buffer, letting the user chain noncontiguous sections into the buffer for later recall.

```
ERAS        LDA   653
            AND   #1
            BNE   ERAS1
            JSR   KILLBUFF
ERAS1       JSR   TOPCLR
            LDA   #<ERASMSG
            LDY   #>ERASMSG
            JSR   PRMSG
ERASAGAIN   LDY   #0
            LDA   (CURR),Y
            EOR   #$80
            STA   (CURR),Y
            JSR   REFRESH
            LDY   #0
            LDA   (CURR),Y
            EOR   #$80
            STA   (CURR),Y
            LDA   #2
            STA   WINDCOLR
            JSR   GETAKEY
            ORA   #64
            CMP   #"W"
            BNE   NOWORD
ERASWORD    JSR   ERA1
            JSR   WRIGHT
            JMP   ERA2
NOWORD      CMP   #"S"
            BNE   UNSENT
ERASENT     JSR   ERA1
            JSR   SRIGHT
            JMP   ERA2
UNSENT      CMP   #"P"
            BNE   NOPAR
            JSR   ERA1
            JSR   PARIGHT
            JMP   ERA2
NOPAR       JSR   CHECK
            JMP   SYSMSG
ERA1        LDA   CURR
            STA   DESTL
            STA   SAVCURR
            LDA   CURR+1
            STA   DESTH
            STA   SAVCURR+1
            RTS
ERA2        SEC
            LDA   CURR
            STA   FROML
            SBC   SAVCURR
            STA   GOBLEN
```

```
        LDA   CURR+1
        STA   FROMH
        SBC   SAVCURR+1
        STA   GOBLEN+1
        JSR   DELC
        LDA   SAVCURR
        STA   CURR
        LDA   SAVCURR+1
        STA   CURR+1
        JSR   REFRESH
        JMP   ERASAGAIN
```

The INPUT routine is used to get responses from the command line. It returns the complete line in INBUFF. INLEN is the length of the input. A zero byte is stored at INBUFF+INLEN after the user presses RETURN. This routine is foolproof (I know...), since no control keys other than DEL are allowed. It also prevents the user from typing past the end of the command line. If the limit of typing length must be set arbitrarily, LIMIT is preset and INPUT is called at INP1. CURSIN is the main loop.

```
INPUT   LDA   #39
        SBC   211
        STA   LIMIT
INP1    LDY   #0
CURSIN  LDA   #153
        JSR   CHROUT
        LDA   #18
        JSR   CHROUT
        LDA   #" "
        JSR   CHROUT
        LDA   #157
        JSR   CHROUT
        STY   INLEN
        JSR   GETAKEY
        LDY   INLEN
        STA   TEMP
        LDA   #146
        JSR   CHROUT
        LDA   #32
        JSR   CHROUT
        LDA   #157
        JSR   CHROUT
        LDA   #155
        JSR   CHROUT
        LDA   TEMP
        CMP   #13
        BEQ   INEXIT
        CMP   #20
        BNE   NOBACK
        DEY
        BPL   NOTZERO
        INY
        JMP   CURSIN
NOTZERO LDA   #157
        JSR   CHROUT
        JMP   CURSIN
NOBACK  LDA   TEMP
        AND   #127
        CMP   #32
        BCC   CURSIN
        CPY   LIMIT
        BEQ   CURSIN
        LDA   TEMP
        STA   INBUFF,Y
        JSR   CHROUT
        LDA   #0
        STA   212
        STA   216
        INY
        JMP   CURSIN
INEXIT  JSR   CHROUT
        LDA   #0
        STA   INBUFF,Y
        TYA
        RTS
```

Here is where most of the input/output routines start. TSAVE saves the entire document area using the Kernal SAVE routine. TOPEN is called by both TSAVE and TLOAD to get the filename and open the file for either tape or disk.

```
TSAVE   JSR   TOPCLR
        LDA   #<SAVMSG
        LDY   #>SAVMSG
        JSR   PRMSG
        JSR   TOPEN
        BCS   ERROR
        LDA   TEXSTART
        STA   TEX
        LDA   TEXSTART+1
        STA   TEX+1
        LDX   LASTLINE
        LDY   LASTLINE+1
        LDA   #TEX
        JSR   SAVE
        BCS   ERROR
```

Location $90 is the value of the Kernal's STatus flag. It's shorter to use LDA $90 than JSR READST.

```
        LDA   $90
        AND   #191
        BNE   ERROR
        JMP   FINE
```

The ERROR message routine. May this routine never be called when you use *SpeedScript*, but that's too much to ask for. The error code from the Kernal routine is 0 if the error was Break Abort. If the device number (DVN) is 8 for disk, we read the disk error channel; otherwise, we just print a generic error message.

```
ERROR     BEQ  STOPPED
          LDA  DVN
          CMP  #8
          BCC  TAPERR
          JSR  READERR
          JMP  ERXIT
TAPERR    LDA  DVN
          CMP  #1
          BEQ  TAPERR
          JSR  TOPCLR
          LDA  #<FNF
          LDY  #>FNF
          JSR  PRMSG
ERXIT     JSR  HIGHLIGHT
          LDA  #1
          STA  MSGFLG
          RTS
STOPPED   JSR  TOPCLR
          LDA  #<BRMSG
          LDY  #>BRMSG
          JSR  PRMSG
          JMP  ERXIT
DVN       .BYT 0
```

TOPEN gets the filename, asks for tape or disk, then calls SETLFS and SETNAM, readying for LOAD or SAVE. If RETURN is pressed without any filename, the return address of the calling routine is pulled off so that we can jump straight back to the MAIN loop.

```
TOPEN     JSR  INPUT
          BEQ  OPABORT
OP2       LDA  #<TDMSG
          LDY  #>TDMSG
          JSR  PRMSG
          JSR  GETAKEY
          LDX  #8
          CMP  #"D"
          BEQ  OPCONT
          LDX  #1
          CMP  #"T"
          BEQ  OPCONT
OPABORT   JSR  SYSMSG
          PLA
          PLA
          RTS
OPCONT    STX  DVN
          LDA  #1
          LDY  #0
          JSR  SETLFS
          LDY  #0
          CPX  #1
          BEQ  SKIPDISK
          LDA  INBUFF,Y
          CMP  #"@"
          NOP
          NOP
          LDA  INBUFF+1,Y
          CMP  #":"
          BEQ  SKIPDISK
          LDA  INBUFF+2,Y
          CMP  #":"
          BEQ  SKIPDISK
```

If **0:**, **1:**, **@0:**, or **xx:** did not precede the filename, we add **0:**. Some think this makes disk writes more reliable. The NOPs above null out the comparison with the @ sign. Originally written as BNE SKIPDISK, this prevented the use of the prefix 1: for owners of dual-drive disk drives.

```
ADDZERO   LDA  #"0"
          STA  FILENAME
          LDA  #":"
          STA  FILENAME+1
COPY1     LDA  INBUFF,Y
          STA  FILENAME+2,Y
          INY
          CPY  INLEN
          BCC  COPY1
          BEQ  COPY1
          INY
          JMP  SETNAME
SKIPDISK  LDA  INBUFF,Y
          STA  FILENAME,Y
          INY
          CPY  INLEN
          BNE  SKIPDISK
SETNAME   STY  FNLEN
          JSR  TOPCLR
          LDA  #<INBUFF
          LDY  #>INBUFF
          JSR  PRMSG
          LDA  FNLEN
          LDX  #<FILENAME
          LDY  #>FILENAME
          JSR  SETNAM
          LDA  #13
          JSR  CHROUT
          JMP  DELITE
```

Called by CTRL-£ to enter a format code. It checks insert mode and inserts if necessary.

```
FORMAT    JSR  TOPCLR
          LDA  #<FORMSG
          LDY  #>FORMSG
          JSR  PRMSG
          JSR  GETAKEY
          JSR  ASTOIN
          ORA  #$80
          PHA
          LDA  INSMODE
          BEQ  NOINS
          JSR  INSCHAR
NOINS     JSR  SYSMSG
          PLA
          JMP  PUTCHR
```

The Load routine checks the cursor position. If the cursor is at the top of text, we call the ERASE routine to wipe out memory before the Load. Otherwise, the Load starts at the cursor position, performing an append.

```
TLOAD   SEC
        LDA   CURR
        SBC   TEXSTART
        STA   TEMP
        LDA   CURR+1
        SBC   TEXSTART+1
        ORA   TEMP
        BEQ   LOAD2
        LDA   #5
        STA   WINDCOLR
LOAD2   JSR   TOPCLR
        LDA   #<LOADMSG
        LDY   #>LOADMSG
        JSR   PRMSG
        JSR   TOPEN
        LDA   WINDCOLR
        CMP   #5
        BEQ   NOER
        JSR   ERASE
NOER    LDA   #0
        LDX   CURR
        LDY   CURR+1
LDVER   JSR   LOAD
        BCC   LOD
        JMP   ERROR
LOD     STX   LASTLINE
        STY   LASTLINE+1
FINE    JSR   CLALL
        JSR   TOPCLR
        LDA   #<OKMSG
        LDY   #>OKMSG
        JSR   PRMSG
        JMP   ERXIT
```

Verify takes advantage of the Kernal routine, so it is very similar to the Load routine.

```
VERIFY  JSR   TOPCLR
        LDA   #<VERMSG
        LDY   #>VERMSG
        JSR   PRMSG
        JSR   TOPEN
        LDA   #1
        LDX   TEXSTART
        LDY   TEXSTART+1
        JSR   LOAD
        LDA   $90
        AND   #191
        BEQ   FINE
        JSR   TOPCLR
        LDA   #<VERERR
        LDY   #>VERERR
        JSR   PRMSG
        JMP   ERXIT
```

DELITE turns off the raster interrupt. You must turn off raster interrupts (and sprites where appropriate) before tape operations. It also restores the default interrupts and fixes the screen colors.

```
DELITE  SEI
        LDA   #0
        STA   $D01A
        STA   53280
        STA   53281
        LDA   #$31
        STA   $314
        LDA   #$EA
        STA   $315
        LDA   #1
        STA   $DC0E
        CLI
        RTS
```

Disk directory routine. It opens "$" as a program file, throws away the link bytes, prints the line number bytes as the blocks used, then prints all following text until the end-of-line zero byte. It's similar to how programs are LISTed in BASIC, except that nothing is untokenized. The system is so sensitive to read errors that we call DCHRIN (which constantly checks for errors) instead of directly calling the Kernal CHRIN routine. DCHRIN can abort the main loop of the DIR routine.

```
CATALOG LDA   #147
        JSR   CHROUT
        LDA   #13
        JSR   CHROUT
        JSR   DELITE
        JSR   DIR
        LDA   #13
        JSR   CHROUT
        LDA   #<DIRMSG
        LDY   #>DIRMSG
        JSR   PRMSG
WAITKEY JSR   GETIN
        CMP   #13
        BNE   WAITKEY
        JSR   HIGHLIGHT
        JMP   SYSMSG
ENDIR   JSR   CLRCHN
        LDA   #1
        JSR   CLOSE
        RTS
DIR     JSR   CLALL
        LDA   #1
        LDX   #8
        LDY   #0
        JSR   SETLFS
        LDA   #1
```

```
          LDX   #<DIRNAME
          LDY   #>DIRNAME
          JSR   SETNAM
          JSR   OPEN
          BCS   ENDIR
          LDX   #1
          JSR   CHKIN
          JSR   DCHRIN
          JSR   DCHRIN
DIRLOOP   JSR   DCHRIN
          JSR   DCHRIN
          BEQ   ENDIR
PAUSE     JSR   CLRCHN
          JSR   GETIN
          CMP   #32
          BNE   NOPAUSE
          JSR   GETAKEY
NOPAUSE   LDX   #1
          JSR   CHKIN
          JSR   DCHRIN
          PHA
          JSR   DCHRIN
          TAY
          PLA
          TAX
          TYA
          LDY   #55
          STY   MAP
          JSR   $BDCD
          LDY   #54
          STY   MAP
          LDA   #32
          JSR   CHROUT
INLOOP    JSR   DCHRIN
          BEQ   DLINE
          JSR   CHROUT
          JMP   INLOOP
DLINE     LDA   #13
          JSR   CHROUT
          JMP   DIRLOOP

DCHRIN    JSR   CHRIN
          PHA
          LDA   $90
          AND   #191
          BEQ   NOSTERR
          PLA
          PLA
          PLA
          JMP   ENDIR
NOSTERR   PLA
          RTS
```

A rather short routine that converts a string of ASCII digits into a number in hex and the accumulator. It takes advantage of decimal mode. In decimal mode, the accumulator is adjusted after additions and subtractions so that it acts like a two-digit decimal counter. We shift BCD over a nybble and add in the left nybble of the ASCII number until we reach the end of the ASCII number. We then subtract 1 from BCD and increment X (which doesn't conform to decimal mode) until BCD is down to zero. The X register magically holds the converted number. Naturally, decimal mode is cleared before this routine exits, or it would wreak major havoc. ASCHEX is used to convert the parameters of printer commands like left margin.

```
ASCHEX    LDX   #0
          STX   BCD
          STX   BCD+1
          STX   HEX
          STX   HEX+1
DIGIT     SEC
          LDA   (TEX),Y
          SBC   #48
          BCC   NONUM
          CMP   #10
          BCS   NONUM
          ASL   BCD
          ROL   BCD+1
          ASL   BCD
          ROL   BCD+1
          ASL   BCD
          ROL   BCD+1
          ASL   BCD
          ROL   BCD+1
          ORA   BCD
          STA   BCD
          INY
          BNE   DIGIT
          INC   TEX+1
          JMP   DIGIT
NONUM     SED
DECHEX    LDA   BCD
          ORA   BCD+1
          BEQ   DONENUM
          SEC
          LDA   BCD
          SBC   #1
          STA   BCD
          LDA   BCD+1
          SBC   #0
          STA   BCD+1
          INC   HEX
          BNE   NOHEXINC
          INC   HEX+1
NOHEXINC  JMP   DECHEX
DONENUM   LDA   HEX
          CLD
          RTS
```

Insert the buffer. This is the recall routine called by CTRL-R. It must not allow an insertion that would overfill memory. It calls DMOVE to open a space in memory, then UMOVE (which is a little faster than DMOVE) to copy the buffer to the empty space.

```
INSBUFFER SEC
          LDA  TPTR
          SBC  TEXBUF
          STA  BUFLEN
          LDA  TPTR+1
          SBC  TEXBUF+1
          STA  BUFLEN+1
          ORA  BUFLEN
          BNE  OKBUFF
          JSR  TOPCLR
          LDA  #<INSMSG
          LDY  #>INSMSG
          JSR  PRMSG
          LDA  #1
          STA  MSGFLG
          RTS
OKBUFF    CLC
          LDA  CURR
          STA  FROML
          ADC  BUFLEN
          STA  DESTL
          LDA  CURR+1
          STA  FROMH
          ADC  BUFLEN+1
          STA  DESTH
          SEC
          LDA  LASTLINE
          SBC  FROML
          STA  LLEN
          LDA  LASTLINE+1
          SBC  FROMH
          STA  HLEN
          CLC
          ADC  DESTH
          CMP  TEXEND+1
          BCC  OKMOV
          JSR  TOPCLR
          LDA  #<INSERR
          LDY  #>INSERR
          JSR  PRMSG
          LDA  #1
          STA  MSGFLG
          RTS
OKMOV     JSR  DMOVE
          CLC
          LDA  BUFLEN
          STA  LLEN
          ADC  LASTLINE
          STA  LASTLINE
          LDA  BUFLEN+1
          STA  HLEN
          ADC  LASTLINE+1
          STA  LASTLINE+1
          LDA  CURR
          STA  DESTL
          LDA  CURR+1
          STA  DESTH
          LDA  TEXBUF
          STA  FROML
          LDA  TEXBUF+1
          STA  FROMH
          LDA  #0
          STA  $D01A
          LDA  #52
          STA  MAP
          JSR  UMOVE
          LDA  #54
          STA  MAP
          LDA  #1
          STA  $D01A
          JMP  CHECK
```

Exchange the character highlighted by the cursor with the character to the right of it. Not a vital command, but it was included due to the brevity of the code.

```
SWITCH    LDY  #0
          LDA  (CURR),Y
          TAX
          INY
          LDA  (CURR),Y
          DEY
          STA  (CURR),Y
          INY
          TXA
          STA  (CURR),Y
          RTS
```

Changes the case of the character highlighted by the cursor.

```
ALPHA     LDY  #0
          LDA  (CURR),Y
          AND  #63
          BEQ  NOTALPHA
          CMP  #27
          BCS  NOTALPHA
          LDA  (CURR),Y
          EOR  #64
          STA  (CURR),Y
NOTALPHA  JMP  RIGHT
```

Converts internal (screen code) format to Commodore ASCII. Used to convert the screen-code format of *SpeedScript* documents to ASCII for the sake of printing.

```
INTOAS    STA  TEMP
          AND  #$3F
          ASL  TEMP
          BIT  TEMP
          BPL  ISK1
          ORA  #$80
ISK1      BVS  ISK2
          ORA  #$40
ISK2      STA  TEMP
          RTS
```

The start of the printer routines. This part could logically be called a separate program, but many variables are common to the above code.

Table of default settings for left margin, right margin, page length, top margin,

bottom margin, etc. See the table starting at LMARGIN at the end of this source code.

```
DEFTAB    .BYT  5,75,66,5,58,1,1,1,0,1,0,8
                0
```

Table of default printer codes.

```
PRCODES   .BYT  27,14,15,18
```

Another advantage of modular coding is that you can change the behavior of a lot of code by just changing one small, common routine. This is a substitute for the Kernal CHROUT, although it calls CHROUT. It checks to see if the current page number equals the page number specified by the user for printing to start. It also checks for the RUN/STOP key to abort the printing and permits printing to be paused with the SHIFT key.

```
PCHROUT     STA   PCR
            TXA
            PHA
            TYA
            PHA
            SEC
            LDA   PAGENUM
            SBC   STARTNUM
            LDA   PAGENUM+1
            SBC   STARTNUM+1
            BCC   SKIPOUT
            LDA   PCR
            JSR   CHROUT
SHIFTFREEZE LDA   653
            AND   #1
            STA   53280
            BNE   SHIFTFREEZE
            LDA   $91
            CMP   #$7F
            BNE   SKIPOUT
            INC   53280
            JSR   CR
            JMP   PEXIT
SKIPOUT     PLA
            TAY
            PLA
            TAX
            LDA   PCR
            RTS
```

Displays "Printing..."

```
PRIN        JSR   TOPCLR
            LDA   #<PRINMSG
            LDY   #>PRINMSG
            JMP   PRMSG
PBORT       JMP   PEXIT
```

Called by CTRL-P. If SHIFT is not held down with CTRL-P, we choose a device number of 4, a secondary address of 7 (lowercase mode), and no filename. If SHIFT is held down, we ask "Print to: Screen, Disk, Printer?" If Screen is selected, we use a device number of 3. If Disk is selected, we get a filename and use a device number and secondary address of 8. For Printer, we ask for the device number and secondary address. SETLFS is called after all these decisions are made, then OPEN. No matter how the file is OPENed, we reference it by file number 1.

```
PRINT       LDA   SCRCOL
            STA   SAVCOL
            LDA   #0
            STA   WINDCOLR
            STA   53280
            STA   SCRCOL
            JSR   SETNAM
            LDA   #4
            STA   DEVNO
            LDY   #7
            LDA   653
            AND   #1
            BNE   ASKQUES
            JMP   OVERQUES
ASKQUES     JSR   TOPCLR
            LDA   #<CHOOSEMSG
            LDY   #>CHOOSEMSG
            JSR   PRMSG
            JSR   GETAKEY
            AND   #127
            LDX   #3
            STX   DEVNO
            CMP   #"S"
            BEQ   PRCONT
NOTSCREEN   LDX   #8
            STX   DEVNO
            CMP   #"D"
            BEQ   DOFN
            CMP   #"P"
            BNE   PBORT
            JSR   TOPCLR
            LDA   #<DEVMSG
            LDY   #>DEVMSG
            JSR   PRMSG
            JSR   GETAKEY
            SEC
            SBC   #48
            CMP   #4
            BCC   PBORT
            CMP   #80
            BCS   PBORT
            STA   DEVNO
            JMP   PRCONT
```

Ask for a print filename, if appropriate, and add ",S,W" for a sequential write file.

```
DOFN      JSR  TOPCLR
          LDA  #<FNMSG
          LDY  #>FNMSG
          JSR  PRMSG
          JSR  INPUT
          BEQ  PBORT
          LDY  INLEN
          LDA  #","
          STA  INBUFF,Y
          INY
          LDA  #"W"
          STA  INBUFF,Y
          INY
          STY  INLEN
          LDA  INLEN
          LDX  #<INBUFF
          LDY  #>INBUFF
          JSR  SETNAM
PRCONT    LDA  DEVNO
          TAY
          CMP  #4
          BCC  OVERQUES
          CMP  #8
          BCS  OVERQUES
NOTD2     JSR  TOPCLR
          LDA  #<SADRMSG
          LDY  #>SADRMSG
          JSR  PRMSG
          JSR  GETAKEY
          SEC
          SBC  #48
          TAY
          BPL  OVERQUES
          JMP  PBORT
OVERQUES  LDA  #1
          LDX  DEVNO
          JSR  SETLFS
          JSR  PRIN
          LDA  #1
          JSR  CLOSE
          JSR  OPEN
          LDX  #1
          JSR  CHKOUT
          BCC  PROK
          JMP  PEXIT
```

Reset several flags (footer length, header length, true ASCII, underline mode, and linefeed mode).

```
PROK      LDX  #0
          STX  FTLEN
          STX  HDLEN
          STX  NEEDASC
          STX  UNDERLINE
          STX  LINEFEED
```

Copy definition tables and default printer codes.

```
COPYDEF   LDA  DEFTAB,X
          STA  LMARGIN,X
          INX
          CPX  #12
          BNE  COPYDEF
          LDA  #$FF
          STA  LINE
          STA  NOMARG
          LDX  #4
COPYDEFS  LDA  PRCODES-1,X
          STA  CODEBUFFER+48,X
          DEX
          BNE  COPYDEFS
```

Reentry point for printing after linked files.

```
RETEX     LDA  TEXSTART
          STA  TEX
          LDA  TEXSTART+1
          STA  TEX+1
```

Main printing loop. We print the left margin, grab a line of text, scan backward until we find a space or a carriage return, then break the line there. If printer codes are encountered, they're passed on to the SPECIAL routine. Otherwise, we end up calling BUFPRT to print the line and process some other control codes.

```
PLOOP     LDY  #0
          STY  POS
          CPY  NOMARG
          BEQ  PLOOP1
          LDA  LMARGIN
          STA  POS
PLOOP1    LDA  (TEX),Y
          BPL  NOTSP
          JMP  SPECIAL
NOTSP     CMP  #RETCHAR
          BEQ  FOUNDSPACE
NOTRET    STA  PRBUFF,Y
          INY
          INC  POS
          LDA  POS
          CMP  RMARGIN
          BCC  PLOOP1
          STY  FINPOS
FINDSPACE LDA  (TEX),Y
          CMP  #32
          BEQ  FOUNDSPACE
          DEC  POS
          DEY
          BNE  FINDSPACE
          LDY  FINPOS
          JMP  OVERSTOR
FSPACE    INY
          LDA  (TEX),Y
          CMP  #32
          BEQ  FOUNDSPACE
          DEY
FOUNDSPAC STY  FINPOS
OVERSTOR  TYA
          SEC
          ADC  TEX
          STA  TEX
          LDA  TEX+1
```

```
            ADC   #0
            STA   TEX+1
            LDY   #0
```

If this is the first page, we need to print the header, if any, with JSR TOP.

```
DOBUFF      LDA   LINE
            CMP   #$FF
            BNE   DOBUF2
            JSR   TOP
DOBUF2      LDA   NOMARG
            BEQ   OVERMARG
            JSR   LMARG
OVERMARG    SEC
            ROL   NOMARG
            LDA   FINPOS
            STA   ENDPOS
            LDA   #<PRBUFF
            STA   INDIR
            LDA   #>PRBUFF
            STA   INDIR+1
            JSR   BUFPRT
```

A line has been printed. We check to see if we've hit the bottom margin and, if so, go to PAGE, which goes to the end of the page, prints the footer (if any), and feeds to the next page.

```
ZBUFF       JSR   CRLF
            LDA   LINE
            CMP   BOTMARG
            BCC   NOTPAGE
            JSR   PAGE
```

Have we reached the end of text?

```
NOTPAGE     SEC
            LDA   TEX
            SBC   LASTLINE
            STA   TEMP
            LDA   TEX+1
            SBC   LASTLINE+1
            ORA   TEMP
            BEQ   DORPT
            BCC   DORPT
```

If so, we check for a footer. If there is one, we set HDLEN and TOPMARG to zero (so that the printhead will end up at the right place on the last page) and call PAGE, which prints the footer. If there is no footer, we leave the printhead on the same page so that paper isn't wasted.

```
            LDA   FTLEN
            BEQ   PXIT
            LDA   #0
            STA   HDLEN
            STA   TOPMARG
            JSR   PAGE
```

Exit routines. If screen output was selected, we wait for a keystroke before going back to editing mode. Since the RUN/STOP key is used to abort printing and to insert a margin indent in editing mode, we wait for the user to let go of RUN/STOP before we return to editing mode.

```
PXIT        LDA   DEVNO
            CMP   #3
            BNE   PEXIT
            JSR   GETAKEY
PEXIT       JSR   STOP
            BEQ   PEXIT
            LDA   #1
            JSR   CLOSE
            JSR   CLALL
            LDA   SAVCOL
            STA   SCRCOL
            LDX   #$FA
            TXS
            JSR   SYSMSG
            JMP   MAIN
DORPT       JMP   PLOOP
```

Paging routines. We skip (PAGELENGTH−LINE)—two blank lines to get to the bottom of the page, print a footer (if there is one) or a blank line (if not), then page to the beginning of the next page, skipping over the paper perforation. If the wait mode is enabled, we wait for the user to insert a new sheet of paper.

```
PAGE        SEC
            LDA   PAGELENGTH
            SBC   LINE
            TAY
            DEY
            DEY
            BEQ   NOSK
            BMI   NOSK
NEXPAGE     JSR   CR
            DEY
            BNE   NEXPAGE
NOSK        LDA   FTLEN
            BEQ   SKIPFT
            STA   ENDPOS
            LDA   #<FTBUFF
            STA   INDIR
            LDA   #>FTBUFF
            STA   INDIR+1
            JSR   LMARG
            JSR   BUFPRT
SKIPFT      JSR   CR
            JSR   CR
            JSR   CR
```

Increment the page number.

```
        INC   PAGENUM
        BNE   NOIPN
        INC   PAGENUM+1
```

The page wait mode is inappropriate when printing to the screen or to disk, or when skipping over pages with the ? format command.

```
NOIPN   LDA   CONTINUOUS
        BNE   TOP
        LDA   DEVNO
        CMP   #3
        BEQ   TOP
        CMP   #8
        BEQ   TOP
        SEC
        LDA   PAGENUM
        SBC   STARTNUM
        LDA   PAGENUM+1
        SBC   STARTNUM+1
        BCC   TOP
        JSR   CLRCHN
        JSR   TOPCLR
        LDA   #<WAITMSG
        LDY   #>WAITMSG
        JSR   PRMSG
        JSR   GETAKEY
        JSR   PRIN
        LDX   #1
        JSR   CHKOUT
```

Print the header, skip to the top margin.

```
TOP     LDA   HDLEN
        BEQ   NOHEADER
        STA   ENDPOS
        LDA   #<HDBUFF
        STA   INDIR
        LDA   #>HDBUFF
        STA   INDIR+1
        JSR   LMARG
        JSR   BUFPRT
NOHEADER LDY  TOPMARG
        STY   LINE
        DEY
        BEQ   SKIPTOP
        BMI   SKIPTOP
TOPLP   JSR   CR
        DEY
        BNE   TOPLP
SKIPTOP RTS
```

Left margin routine. This routine is not called if NOMARG is selected (margin release).

```
LMARG   LDA   #32
        LDY   LMARGIN
        STY   POS
        BEQ   LMEXIT
LMLOOP  JSR   PCHROUT
        DEY
        BNE   LMLOOP
LMEXIT  RTS
```

CRLF is called at the end of most printed lines. It increments the LINE count and takes into account the current line spacing mode set by the s format command.

```
CRLF    LDY   SPACING
        CLC
        TYA
        ADC   LINE
        STA   LINE
CRLOOP  JSR   CR
        DEY
        BNE   CRLOOP
        RTS
```

CR just prints a single carriage return and linefeed (if specified).

```
CR      LDA   #13
        JSR   PCHROUT
        LDA   LINEFEED
        BEQ   NOLF
        JSR   PCHROUT
NOLF    RTS
```

Handle special printer codes like left margin. This looks up the printer code using a routine similar to CONTROL.

```
SPECIAL STA   SAVCHAR
        AND   #127
        JSR   INTOAS
        LDX   SPTAB
SRCHSP  CMP   SPTAB,X
        BEQ   FSP
        DEX
        BNE   SRCHSP
        DEC   POS
        JMP   DEFINE
FSP     DEX
        TXA
        ASL
        TAX
        STY   YSAVE
        LDA   #>SPCONT-1
        PHA
        LDA   #<SPCONT-1
        PHA
        LDA   SPVECT+1,X
        PHA
        LDA   SPVECT,X
        PHA
        RTS
```

After the format code is processed, we must skip over the format command and its parameter so that it's not printed.

```
SPCONT  SEC
        LDA  YSAVE
        ADC  TEX
        STA  TEX
        LDA  TEX+1
        ADC  #0
        STA  TEX+1
        JMP  PLOOP
```

If the format command ends with a return mark, we must skip over the return mark as well.

```
SPCEXIT LDA  (TEX),Y
        CMP  #RETCHAR
        BEQ  NOAD
        DEY
NOAD    STY  YSAVE
        RTS
```

Special format code table. It starts with the number of format commands, then the characters for each format command.

```
SPTAB   .BYT  18
        .ASC  "WALRTBSNHF@P?
              XMIGJ"
```

The *address*−*1* of each format routine.

```
SPVECT  .WOR  PW−1,AS−1,LM−1,
              RM−1,TP−1
        .WOR  BT−1,SP−1,NX−1,H
              D−1,FT−1
        .WOR  PN−1,PL−1,SPAGE
              −1,ACROSS−1
        .WOR  MRELEASE−1,COM
              MENT−1,LINK−1
        .WOR  LFSET−1
```

m Margin release. INY is used to skip over the format character.

```
MRELEASE INY
         LDA  #0
         STA  NOMARG
         JMP  SPCEXIT
```

x Columns across, used by centering.

```
ACROSS  INY
        JSR  ASCHEX
        STA  PAGEWIDTH
        JMP  SPCEXIT
```

? Start printing at specified page.

```
SPAGE   INY
        JSR  ASCHEX
        STA  STARTNUM
        LDA  HEX+1
        STA  STARTNUM+1
        JMP  SPCEXIT
```

@ Set starting default page number.

```
PN      INY
        JSR  ASCHEX
        STA  PAGENUM
        LDA  HEX+1
        STA  PAGENUM+1
        JMP  SPCEXIT
```

p Page length.

```
PL      INY
        JSR  ASCHEX
        STA  PAGELENGTH
        JMP  SPCEXIT
```

w Set page wait mode.

```
PW      LDA  #0
        STA  CONTINUOUS
        INY
        JMP  SPCEXIT
```

j Set linefeed mode.

```
LFSET   LDA  #10
        STA  LINEFEED
        INY
        JMP  SPCEXIT
```

a Set true ASCII mode.

```
AS      INY
        LDA  #1
        STA  NEEDASC
        JMP  SPCEXIT
```

l Left margin.

```
LM      INY
        JSR  ASCHEX
        STA  LMARGIN
        JMP  SPCEXIT
```

r Right margin.

```
RM      INY
        JSR  ASCHEX
        STA  RMARGIN
        JMP  SPCEXIT
```

t Top margin.

```
TP      INY
        JSR  ASCHEX
        STA  TOPMARG
        JMP  SPCEXIT
```

b Bottom margin.

```
BT      INY
        JSR  ASCHEX
        STA  BOTMARG
        JMP  SPCEXIT
```

s Set line spacing.

```
SP      INY
        JSR  ASCHEX
        STA  SPACING
        JMP  SPCEXIT
```

n Jump to next page.

```
NX          LDY   YSAVE
            INY
            TYA
            PHA
            JSR   PAGE
            PLA
            TAY
            STY   YSAVE
            RTS
```

h Define header. Copy header into header buffer.

```
HD          JSR   PASTRET
            DEY
            STY   HDLEN
            LDY   #1
HDCOPY      LDA   (TEX),Y
            STA   HDBUFF-1,Y
            INY
            CPY   HDLEN
            BCC   HDCOPY
            BEQ   HDCOPY
            INY
            JMP   SPCEXIT
```

Skip just past the return mark.

```
PASTRET     INY
            LDA   (TEX),Y
            CMP   #RETCHAR
            BNE   PASTRET
            RTS
```

f Define footer.

```
FT          JSR   PASTRET
            DEY
            STY   FTLEN
            LDY   #1
FTCOPY      LDA   (TEX),Y
            STA   FTBUFF-1,Y
            INY
            CPY   FTLEN
            BCC   FTCOPY
            BEQ   FTCOPY
            JMP   SPCEXIT
```

i Ignore a line of information.

```
COMMENT     JSR   PASTRET
            JMP   SPCEXIT
```

Define programmable printkeys. We check for =. If not found, this is not an assignment, so we just skip past the code. Otherwise, we use the screen code value as the index into the CODEBUFFER and put the value there, ready to be called during printing by BUFPRT.

```
DEFINE      INY
            LDA   (TEX),Y
            CMP   #"="
            BEQ   DODEFINE
            DEY
            LDA   SAVCHAR
            JMP   NOTRET
DODEFINE    INY
            JSR   ASCHEX
            PHA
            LDA   SAVCHAR
            AND   #127
            TAX
            PLA
            STA   CODEBUFFER,X
            JSR   SPCEXIT
            JMP   SPCONT
```

Link to next file. The filename is called from text; we check for T or D to get the proper device number, erase the text in memory, then call the Kernal Load routine. After the load, we check for a load error, then jump to RETEX to continue printing.

```
LINK        INY
            LDX   #8
            LDA   (TEX),Y
            AND   #63
            CMP   #"D"-64
            BEQ   LINK2
            LDX   #1
            CMP   #"T"-64
            BEQ   LINK2
            JMP   PBORT
LINK2       STX   DVN
            INY
            LDA   (TEX),Y
            CMP   #":"
            BEQ   LINKLOOP
            JMP   PBORT
LINKLOOP    INY
            LDA   (TEX),Y
            CMP   #RETCHAR
            BEQ   OUTNAM
            JSR   INTOAS
            STA   FILENAME-3,Y
            JMP   LINKLOOP

OUTNAM      TYA
            SEC
            SBC   #3
            LDX   #<FILENAME
            LDY   #>FILENAME
            JSR   SETNAM
            JSR   CLRCHN
            LDA   #2
            JSR   CLOSE
            LDA   #2
            LDX   DVN
            LDY   #0
            JSR   SETLFS
            JSR   ERASE
            LDA   #0
            LDX   CURR
            LDY   CURR+1
```

```
        JSR   LOAD
        BCC   OKLOD
        JMP   PBORT
OKLOD   STX   LASTLINE
        STY   LASTLINE+1
        PLA
        PLA
        LDX   #1
        JSR   CHKOUT
        JMP   RETEX
```

Not a printer command. DCMND calls INPUT for a disk command. If RETURN is pressed without a disk command, we jump straight to displaying the disk error message. Otherwise, we send the command and fall through to checking the disk error message to let the user know the success of the command.

```
DCMND   JSR   CLALL
        LDA   #0
        JSR   SETNAM
        LDA   #15
        LDX   #8
        LDY   #15
        JSR   SETLFS
        JSR   OPEN
        BCC   OKD
DCOUT   LDA   #15
        JSR   CLOSE
        JSR   CLALL
        JMP   SYSMSG
OKD     JSR   TOPCLR
        LDA   #<DCMSG
        LDY   #>DCMSG
        JSR   PRMSG
        JSR   INPUT
        BEQ   READERR
        LDX   #15
        JSR   CHKOUT
        BCS   DCOUT
        LDA   #<INBUFF
        LDY   #>INBUFF
        JSR   PRMSG
        LDA   #13
        JSR   CHROUT
        JSR   CLRCHN
```

READERR is called by DCMND and the ERROR routine. It does a CHKIN, then calls INPUT, which automatically displays the message. CLRCHN cleans it up, and we're through.

```
READERR JSR   CLALL
        LDA   #0
        JSR   SETNAM
        LDA   #15
        LDX   #8
        LDY   #15
        JSR   SETLFS
        JSR   OPEN
        BCS   DCOUT
        JSR   TOPCLR
        LDX   #15
        JSR   CHKIN
        JSR   INPUT
        JSR   CLRCHN
        LDA   #15
        JSR   CLOSE
        JSR   CLALL
        LDA   #1
        STA   MSGFLG
        RTS
```

Global search and replace. This just links together the search-specify routine, the replace-specify routine, then repeatedly calls Hunt and Replace, until Hunt returns "Not Found." (FPOS+1 is $FF after a search failure.)

```
SANDR   JSR   RESET
        LDA   HUNTLEN
        BEQ   NOSR
        JSR   ASKREP
SNR     JSR   CONTSRCH
        LDA   FPOS+1
        CMP   #$FF
        BEQ   NOSR
        JSR   REPL
        JSR   REFRESH
        JMP   SNR
NOSR    JMP   SYSMSG
```

If SHIFT is held down, we ask for and store the hunt phrase. If SHIFT is not down, we perform the actual hunt. The line in the INBUFF is compared with characters in text. If at any point the search fails, we continue the comparison with the first character of INBUFF. The search is a failure if we reach the end-of-text. If the entire length of INBUFF matches, the search succeeds, so we change the CURRent cursor position to the found position, save the found position for the sake of the replace routine, then call CHECK to scroll to the found position.

```
HUNT    LDA   653
        CMP   #5
        BNE   CONTSRCH
RESET   JSR   TOPCLR
        LDA   #<SRCHMSG
        LDY   #>SRCHMSG
        JSR   PRMSG
        JSR   INPUT
        STA   HUNTLEN
        BNE   OKSRCH
        JMP   SYSMSG
```

```
OKSRCH    LDY  #0
TOBUFF    LDA  INBUFF,Y
          STA  HUNTBUFF,Y
          INY
          CPY  INLEN
          BNE  TOBUFF
          JMP  SYSMSG
CONTSRCH  LDA  CURR
          STA  TEX
          LDA  CURR+1
          STA  TEX+1
          LDA  #$FF
          STA  FPOS+1
          LDY  #1
          LDX  #0
          LDA  HUNTLEN
          BEQ  NOTFOUND
SRCH1     LDA  HUNTBUFF,X
          JSR  ASTOIN
          CMP  (TEX),Y
          BEQ  CY
          LDX  #$FF
CY        INY
          BNE  NOVFL
          INC  TEX+1
          LDA  TEX+1
          CMP  LASTLINE+1
          BEQ  NOVFL
          BCS  NOTFOUND
NOVFL     INX
          CPX  HUNTLEN
          BNE  SRCH1
          CLC
          TYA
          ADC  TEX
          STA  TEMP
          LDA  TEX+1
          ADC  #0
          STA  TEMP+1
          LDA  LASTLINE
          CMP  TEMP
          LDA  LASTLINE+1
          SBC  TEMP+1
          BCC  NOTFOUND
          SEC
          LDA  TEMP
          SBC  HUNTLEN
          STA  CURR
          STA  FPOS
          LDA  TEMP+1
          SBC  #0
          STA  CURR+1
          STA  FPOS+1
          JSR  CHECK
          RTS
NOTFOUND  JSR  TOPCLR
          LDA  #<NFMSG
          LDY  #>NFMSG
          JSR  PRMSG
          LDA  #1
          STA  MSGFLG
          RTS
```

The replace routine checks to see if SHIFT is held down. If it is, we ask for a replace phrase, and exit. If not, we check to see if the cursor is at the position previously located by the search routine. If it is, we delete the found phrase, then insert the replace phrase. The cursor is moved past the replace phrase for the sake of the next search. This also prevents endless recursion, as in replacing *in* with *winner.*

```
REPSTART  LDA  653
          CMP  #5
          BNE  REPL
ASKREP    JSR  TOPCLR
          LDA  #<REPMSG
          LDY  #>REPMSG
          JSR  PRMSG
          JSR  INPUT
          STA  REPLEN
          BEQ  NOREP
          LDY  #0
REPMOV    LDA  INBUFF,Y
          STA  REPBUFF,Y
          INY
          CPY  INLEN
          BNE  REPMOV
NOREP     JMP  SYSMSG
REPL      SEC
          LDA  CURR
          STA  DESTL
          SBC  FPOS
          STA  TEMP
          LDA  CURR+1
          STA  DESTH
          SBC  FPOS+1
          ORA  TEMP
          BNE  NOREPL
          LDA  #$FF
          STA  FPOS+1
          CLC
          LDA  HUNTLEN
          ADC  CURR
          STA  FROML
          LDA  #0
          ADC  CURR+1
          STA  FROMH
          SEC
          LDA  LASTLINE
          SBC  DESTL
          STA  LLEN
          LDA  LASTLINE+1
          SBC  DESTH
          STA  HLEN
          JSR  UMOVE
          SEC
          LDA  LASTLINE
          SBC  HUNTLEN
          STA  LASTLINE
          LDA  LASTLINE+1
```

```
            SBC   #0
            STA   LASTLINE+1
            LDA   REPLEN
            BEQ   NOREPL
            STA   INSLEN
            LDA   #0
            STA   INSLEN+1
            JSR   INSBLOCK
            LDY   #0
REPLOOP     LDA   REPBUFF,Y
            JSR   ASTOIN
            STA   (CURR),Y
            INY
            CPY   REPLEN
            BNE   REPLOOP
            CLC
            LDA   CURR
            ADC   REPLEN
            STA   CURR
            LDA   CURR+1
            ADC   #0
            STA   CURR+1
NOREPL      JMP   CHECK
```

Suddenly we're back to a PRINT subroutine. This examines the buffer as it's being printed, checking for printkeys and Stage 2 commands like centering.

```
BUFPRT      LDY   #0
BUFLP       CPY   ENDPOS
            BEQ   ENDBUFF
            LDA   (INDIR),Y
            BMI   SPEC2
            JSR   INTOAS
            JSR   CONVASC
            JSR   PCHROUT
```

In underline mode, after we print the character, we backspace the printhead and print an underline character.

```
            LDA   UNDERLINE
            BEQ   NOBRK
            LDA   #8
            JSR   PCHROUT
            LDA   #95
            JSR   PCHROUT
NOBRK       INY
            JMP   BUFLP
ENDBUFF     RTS
```

Stage 2 format commands.

```
SPEC2       STY   YSAVE
            AND   #127
            STA   SAVCHAR
            JSR   INTOAS
```

Centering looks at the length of the line, then sends out extra spaces (the left margin has already been printed) to move the printhead to the right place.

```
OTHER       CMP   #"C"
            BNE   NOTCENTER
            SEC
            LDA   PAGEWIDTH
            SBC   ENDPOS
            LSR
            SEC
            SBC   LMARGIN
            TAY
            LDA   #32
CLOOP       JSR   PCHROUT
            DEY
            BNE   CLOOP
            LDY   YSAVE
            JMP   NOBRK
```

Edge right. This subtracts the length of the line from the right-margin position and moves the printhead to this position. The BUFPRT loops finishes the line.

```
NOTCENTER   CMP   #"E"
            BNE   NOTEDGE
EDGE        SEC
            LDA   RMARGIN
            SBC   ENDPOS
            SEC
            SBC   LMARGIN
            TAY
            LDA   #32
            JMP   CLOOP
```

Toggle underline mode.

```
NOTEDGE     CMP   #"U"
            BNE   NOTOG
            LDA   UNDERLINE
            EOR   #1
            STA   UNDERLINE
```

Substitute the current page number for the # symbol.

```
NOTOG       CMP   #"#"
            BNE   DOCODES
DOPGN       STY   YSAVE
            LDX   PAGENUM
            LDA   PAGENUM+1
            LDY   #55
            STY   MAP
            JSR   $BDCD
            LDY   #54
            STY   MAP
            LDY   YSAVE
            JMP   NOBRK
```

Do special format codes. This just uses the screen-code value of the character as an index into the CODEBUFFER, then sends out the code. *SpeedScript* makes no judgment on the code being sent out.

```
DOCODES  LDX  SAVCHAR
         LDA  CODEBUFFER,X
         JSR  PCHROUT
         JMP  NOBRK
```

This checks for true ASCII mode and, if enabled, exchanges uppercase and lowercase. Used for certain non-Commodore printers and interfaces.

```
CONVASC  LDX  NEEDASC
         BEQ  SKIPASC
         STA  TEMP
         AND  #127
         CMP  #"A"
         BCC  SKIPASC
         CMP  #"Z"+1
         BCS  SKIPASC
         TAX
         LDA  TEMP
         AND  #128
         EOR  #128
         LSR
         LSR
         STA  TEMP
         TXA
         ORA  TEMP
SKIPASC  RTS
```

Display free memory.

```
FREEMEM  JSR  TOPCLR
         SEC
         LDA  TEXEND
         SBC  LASTLINE
         TAX
         LDA  TEXEND+1
         SBC  LASTLINE+1
         LDY  #55
         STY  MAP
         JSR  $BDCD
         LDY  #54
         STY  MAP
         LDA  #1
         STA  MSGFLG
         RTS
```

The message table should be typed in the lowercase mode (SHIFT–Commodore key).

```
msg1     .byt  8,14,155,146
         .asc  "SpeedScript 3.1"
         .byt  0
msg2     .asc  " by Charles
               Brannon"
         .byt  0
killmsg  .asc  "Buffer Cleared"
         .byt  0
buferr   .asc  "Buffer Full"
         .byt  0
delmsg   .asc  "Delete (S,W,P)"
         .byt  0
ymsg     .asc  ": Are you sure?
               (Y/N):"
         .byt  0
clrmsg   .asc  "ERASE ALL TEXT"
         .byt  0
erasmsg  .asc  "Erase (S,W,P): "
         .byt  18
         .asc  "RETURN"
         .byt  146
         .asc  " to exit"
         .byt  0
formsg   .asc  "Press format key:"
         .byt  0
savmsg   .asc  "Save:"
         .byt  0
fnf      .asc  "Tape ERROR"
         .byt  0
brmsg    .asc  "Stopped"
         .byt  0
vererr   .asc  "Verify Error"
         .byt  0
okmsg    .asc  "No errors"
         .byt  0
tdmsg    .byt  147,32,18,212,146
         .asc  "ape or "
         .byt  18,196,146
         .asc  "isk?"
         .byt  0
loadmsg  .asc  "Load:"
         .byt  0
vermsg   .asc  "Verify:"
         .byt  0
dirmsg   .asc  "Press "
         .byt  18
         .asc  "RETURN"
         .byt  146,0
dcmsg    .asc  "Disk command:"
         .byte 0
dirname  .asc  "$"
inserr   .asc  "No Room"
         .byt  0
insmsg   .asc  "No text in buffer."
         .byt  0
choosemsg .byt 147
         .asc  "Print to: "
         .byt  18,211,146
         .asc  "creen,"
         .byt  18,196,146
         .asc  "isk,"
         .byt  18,208,146
         .asc  "rinter?"
         .byt  0
devmsg   .asc  "Device number?"
         .byt  0
sadrmsg  .asc  "Secondary Address
               #?"
         .byt  0
fnmsg    .asc  "Print to filename:"
         .byte 0
prinmsg  .byt  147
         .asc  "Printing..."
         .byt  13,13,0
waitmsg  .asc  "Insert next sheet,
               press "
         .byt  18
```

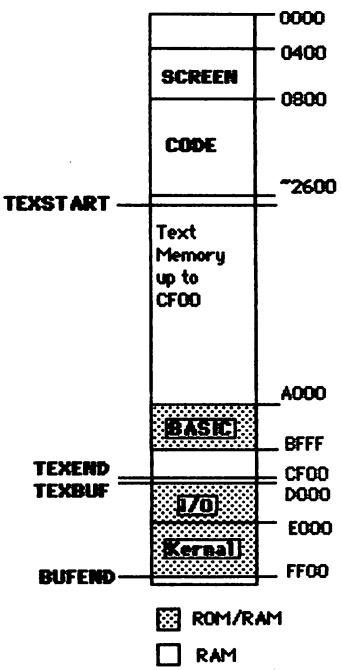
```
            .asc    "RETURN"
            .byt    146,0
srchmsg     .asc    "Hunt for:"
            .byt    0
nfmsg       .asc    "Not Found"
            .byt    0
repmsg      .asc    "Replace with:"
            .byt    0
xitmsg      .asc    "EXIT SpeedScript"
            .byt    0
```

Most variables are here at the end.
They do not become part of the object code.

```
TEXSTART    *=    *+2     ;Start of text area
TEXEND      *=    *+2     ;End of text area
TEXBUF      *=    *+2     ;Start of buffer
BUFEND      *=    *+2     ;End of buffer area
LENTABLE    *=    *+1     ;Length of first screen line
TOPLIN      *=    *+2     ;Home position in text
MSGFLG      *=    *+1     ;Message flag
INSMODE     *=    *+1     ;Insert mode
ENDPOS      *=    *+1     ;Used by delete routines
FINPOS      *=    *+1     ;" "
LASTLINE    *=    *+2     ;End-of-text position
LIMIT       *=    *+1     ;Used by INPUT
INLEN       *=    *+1     ;" "
BOTSCR      *=    *+2     ;Bottom of screen in text
LBUFF       *=    *+40    ;Line buffer (REFRESH)
INBUFF      *=    *+40    ;INPUT buffer
FILENAME    *=    *+24    ;Stores filename
FNLEN       *=    *+1     ;Length of filename
SAVCURR     *=    *+2     ;Used by delete routines
BCD         *=    *+2     ;Used by ASCHEX
HEX         *=    *+2     ;" "
TPTR        *=    *+2     ;Last character in buffer
BUFLEN      *=    *+2     ;Buffer length
GOBLEN      *=    *+2     ;Size of deleted text
FROMSAV     *=    *+2     ;Used by delete routines
DESTSAV     *=    *+2     ;" "
HDLEN       *=    *+1     ;Header length
FTLEN       *=    *+1     ;Footer length
LMARGIN     *=    *+1     ;Holds left margin
RMARGIN     *=    *+1     ;Right margin
PAGELENGTH  *=    *+1     ;Page length
TOPMARG     *=    *+1     ;Top margin
BOTMARG     *=    *+1     ;Bottom margin
SPACING     *=    *+1     ;Line spacing
CONTINUOUS  *=    *+1     ;Page wait mode
PAGENUM     *=    *+2     ;Page number
STARTNUM    *=    *+2     ;Start printing at #
PAGEWIDTH   *=    *+1     ;Columns across
NOMARG      *=    *+1     ;Margin release flag
POS         *=    *+1     ;POSition within line
LINE        *=    *+1     ;Line count
YSAVE       *=    *+1     ;Preserves Y register
SAVCHAR     *=    *+1     ;Preserves accumulator
INSLEN      *=    *+1     ;Length of an insertion
DEVNO       *=    *+1     ;Device number
NEEDASC     *=    *+1     ;True ASCII flag
UNDERLINE   *=    *+1     ;Underline mode flag
FPOS        *=    *+2     ;Found position
```

```
PCR          *=    *+1     ;Used by PCHROUT
HUNTLEN      *=    *+1     ;Length of hunt phrase
HUNTBUFF     *=    *+30    ;Holds hunt phrase
REPLEN       *=    *+1     ;Length of replace phrase
REPBUFF      *=    *+30    ;Holds replace phrase
CODEBUFFER   *=    *+128   ;Holds definable printkeys
PRBUFF       *=    *+256   ;Printer line buffer
HDBUFF       *=    *+256   ;Holds header
FIRSTRUN     *=    *+1     ;Has program been run before?
FTBUFF       *=    *+256   ;Holds footer
SAVCOL       *=    *+1     ;Save SCRCOL
LINEFEED     *=    *+1     ;Linefeed mode flag
BLINKFLAG    *=    *+1     ;Is cursor in blink phase?
END          .END           ;+$100 is TEXSTART
```

Figure 3-1. SpeedScript Memory Map



Appendices

ScriptSave
Automatic Disk SAVEs for Commodore 64 *SpeedScript 3.1*

J. Blake Lambert

Have you ever watched your computer suddenly blink off due to an unexpected power failure and then realized that you haven't saved your text for an hour or more? All that work down the drain. But with "ScriptSave" these accidents won't be quite so disastrous. The machine language program is designed to work with a Commodore 64, a disk drive, and the Commodore 64 version of SpeedScript 3.1.

While you are working with a computer, you're tethered to a lifeline. That lifeline is the computer's power cord. If the lifeline is disconnected or interrupted for even a brief moment, your computer suffers an attack of amnesia. Random Access Memory (RAM) chips need a constant flow of electricity to maintain their information—the information you put into the computer. Usually, a power failure does not damage the computer, but it does obliterate the program or text you were working on.

Luckily, most people live in areas with reliable power sources. However, electrical service in some locales is subject to frequent interruptions. And sometimes your wayward foot, a passerby, a small child, or even a pet can accidentally knock a power cord loose. A split second is all it takes for the computer to forget.

Unfortunately, the writer is often forgetful, too. To protect yourself against power interruptions, you should periodically save your work on disk. But when you're working intensely, it's easy to forget this important duty. If the power does fail, you can generally remember where you left off, but it's often impossible to remember how you got there. Even if you frequently rewrite your documents, losing any of the intermediate versions interferes with the creative process.

An Extra Rope

"ScriptSave" is the solution. ScriptSave is a short (less than 256-byte) utility that ties into the Commodore 64 version of

SpeedScript 3.1. Every ten minutes, it waits for you to finish the paragraph you're working on, then it automatically saves your text (except for the final return mark character) on disk with a special filename. That way, if a power failure unexpectedly strikes, you can later recover all but the last ten minutes of your work.

ScriptSave is a BASIC loader and boot program: It loads and executes *both* the machine language automatic-save routine and *SpeedScript 3.1.* Before running ScriptSave, save it to disk. Since *SpeedScript* loads into the same area of memory as the ScriptSave loader, the loader is erased each time it is run. Make sure both programs are on the same disk, and change the filename in line 30 of ScriptSave (listed below as "SS3") to the filename for *SpeedScript 3.1* as stored on your disk. Generally, it is best to start with a blank disk and place ScriptSave on the disk first. This way, you can use LOAD"*",8: followed by SHIFT-RUN/STOP to boot up for a writing session.

Once you load and run ScriptSave, this prompt should appear:

File:

Type in a legal Commodore filename, but limit it to 14 characters or less. Press RETURN. ScriptSave automatically loads and runs *SpeedScript 3.1.* Now you can start writing and stop worrying about periodic SAVEs.

When ScriptSave stores your work, it precedes the filename you specified with a two-digit version number. For example, if you choose the filename ARTICLE, the first version will be called 01ARTICLE, the second version 02ARTICLE, and so on.

Of course, you can still save manually anytime you wish. *SpeedScript 3.1* functions normally except for one detail—it assumes that all your SAVEs are on disk. You no longer have to press T or D to specify Tape or Disk after selecting the f8 SAVE option.

Additional Notes

When you finish writing, you will probably want to save a final version of your text. Later, if you want to scratch the intermediate versions off your disk, there's a quick method using *SpeedScript*'s disk commands. First, press CTRL–up arrow (↑). When *SpeedScript* prompts "Disk Command:", type s:??*article*

and press RETURN (substitute for *article* the filename you specified in ScriptSave).

There's another trick you can use to give yourself more time between SAVEs or to force an early SAVE. Since ScriptSave uses the internal time-of-day clock, you can exit *SpeedScript 3.1* by tapping RESTORE and pressing the Y key, and then from BASIC, type POKE 56330,0 to reset the timer and delay the SAVE. Or you can POKE 56330,16 to set the timer for an immediate SAVE, which will be activated the next time you press RETURN while in *SpeedScript*. You can toggle ScriptSave off and on by entering SYS 52993. Each time ScriptSave is toggled on, it resets the version number to 01 and prompts you to enter a new filename. All of these commands (except for toggling ScriptSave on) should be followed by RUN to reenter *SpeedScript*. One caution, however: While these manipulations are usually safe, there is a chance that exiting and reentering *SpeedScript* will erase your text.

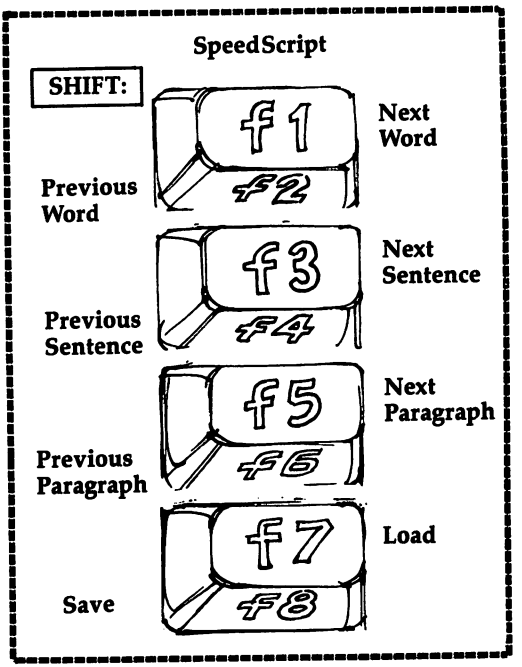
Program A-1. ScriptSave

For mistake-proof program entry, be sure to read "The Automatic Proofreader," in Chapter 2.

```
10 FOR I=52993 TO 53246:READ A:CK=CK+A:POKE I,A:NE
   XT                                          :rem 175
20 IF CK<>29572 THEN PRINT"{RVS}ERROR IN DATA STAT
   EMENTS":STOP                                :rem 215
30 PRINT"{CLR}LOAD"CHR$(34)"SS3"CHR$(34)",8"
                                                :rem 30
40 PRINT"{4 DOWN}SYS52993"                     :rem 132
50 POKE631,19:POKE632,13:POKE633,13:POKE198,3:END
                                               :rem 104
52993 DATA 173,236,2,73,1,141                  :rem 155
52999 DATA 236,2,208,12,160,2                  :rem 155
53005 DATA 185,252,207,153,189,10               :rem 95
53011 DATA 136,16,247,96,160,3                 :rem 199
53017 DATA 185,248,207,153,33,19                :rem 52
53023 DATA 136,16,247,169,48,141                :rem 51
53029 DATA 167,2,141,168,2,169                 :rem 208
53035 DATA 212,160,207,32,30,171                :rem 28
53041 DATA 169,227,160,207,32,30                :rem 38
53047 DATA 171,32,0,172,160,2                  :rem 138
53053 DATA 185,254,1,153,167,2                 :rem 201
53059 DATA 240,3,200,208,245,140                :rem 33
53065 DATA 237,2,169,32,141,189                  :rem 4
53071 DATA 10,169,112,141,190,10                :rem 28
53077 DATA 169,207,141,191,10,32                :rem 44
53083 DATA 96,207,76,13,8,160                  :rem 161
53089 DATA 1,140,238,2,136,140                 :rem 195
```

```
53095 DATA 8,220,140,9,220,140          :rem 191
53101 DATA 10,220,96,138,201,13         :rem 231
53107 DATA 240,9,201,141,240,7          :rem 187
53113 DATA 104,104,76,196,10,162         :rem 36
53119 DATA 95,142,239,2,173,10          :rem 202
53125 DATA 220,41,240,240,70,206         :rem 26
53131 DATA 238,2,208,214,238,168         :rem 47
53137 DATA 2,173,168,2,201,58           :rem 151
53143 DATA 208,20,169,48,141,168         :rem 51
53149 DATA 2,238,167,2,173,167          :rem 212
53155 DATA 2,201,58,208,5,169           :rem 154
53161 DATA 48,141,167,2,169,214         :rem 255
53167 DATA 160,207,32,113,9,173         :rem 250
53173 DATA 237,2,162,167,160,2          :rem 199
53179 DATA 32,189,255,169,1,162          :rem 12
53185 DATA 8,160,0,142,27,19            :rem 101
53191 DATA 32,186,255,32,197,18            :rem 8
53197 DATA 32,96,207,174,239,2          :rem 216
53203 DATA 96,18,14,147,211,67          :rem 205
53209 DATA 82,73,80,84,211,65           :rem 160
53215 DATA 86,69,146,0,32,194           :rem 161
53221 DATA 76,65,75,69,32,204           :rem 161
53227 DATA 65,77,66,69,82,84            :rem 131
53233 DATA 13,198,73,76,69,58           :rem 175
53239 DATA 0,162,8,208,24,138           :rem 153
53245 DATA 201,13                        :rem 64
```

Clip-Out Function-Key Overlay



Clip-Out Quick-Reference Card—Editing Commands

CTRL A	Change case
CTRL B	Change border color
CTRL D	Delete (Sentence, Word, Paragraph)
CTRL E	Erase (Sentence, Word, Paragraph)
CTRL G	Global search and replace
CTRL H	Hunt for phrase with SHIFT: Select hunt phrase
CTRL I	Enter/exit insert mode
CTRL J	Replace with SHIFT: Select replace phrase
CTRL K	Kill buffer
CTRL L	Change text character color
CTRL P	Print
CTRL R	Restore buffer
CTRL V	Verify
CTRL X	Transpose characters
CTRL Z	Go to end of text
CTRL =	Display amount of free memory
CTRL ↑	Send disk command or read error channel
CTRL 4	Display disk directory
CTRL £	Enter format (printer) commands
CTRL 3	Commodore 64 only: Same as CTRL-£
CLR/HOME	Press once to go to top of screen Hold down to go to top of text with SHIFT: Erase all text
CRSR (left/right)	Move the cursor left one character with SHIFT: Move the cursor right one character
CRSR (up/down)	Go to next sentence with SHIFT: Go to previous sentence
RUN/STOP	Indent 5 spaces with SHIFT: Insert 255 spaces
RESTORE	Exit *SpeedScript* (Commodore 64) with RUN/STOP: Exit *SpeedScript* (VIC-20)
←	Backspace with CTRL: Delete character under cursor and close up text with SHIFT and CTRL: Delete all spaces from cursor to next character
RETURN	Return mark with SHIFT: End paragraph, add an extra return mark, and indent next paragraph
INST/DEL	Delete character with SHIFT: Insert space

Clip-Out Quick-Reference Card—Format Commands

Enter these commands with CTRL-£ (on the Commodore 64, you can alternatively use CTRL-3):

Command	Description	Default
a	True ASCII	off
b	Bottom Margin	58
c	Centering	
e	Edge Right	
f	Footer	
g	Goto Linked File*	
h	Header	
i	Information*	
j	Select linefeeds*	
l	Left Margin	5
m	Margin Release*	
n	Next Page	
p	Page Length*	66
r	Right Margin	75
s	Spacing	1
t	Top Margin	5
u	Underline toggle	
w	Page Wait	
x	Columns across*	80
@	Initial page #*	1
?	Skip pages*	
#	Print page number	

hc SpeedScript/#←	Centered Header with page number
l10r70s2←	Left margin 10, right margin 70, double spacing.
gD:SpeedScript.2←	Goto and continue printing with filename "SpeedScript.2"

* Notes command changed or added since Version 2.0

Index

Notes

Notes

To order your copy of the Commodore *SpeedScript* Disk call our toll-free US order line: 1-800-334-0868 (in NC call 919-275-9809) or send your prepaid order to:

Commodore *SpeedScript* Disk
COMPUTE! Publications
P.O. Box 5058
Greensboro, NC 27403

All orders must be prepaid (check, charge, or money order). NC residents add 4.5% sales tax.

Send _____ copies of the Commodore *SpeedScript* Disk at $12.95 per copy.

Subtotal $_____

Shipping & Handling: $2.00/disk* $_____

Sales tax (if applicable) $_____

Total payment enclosed $_____

*Outside US and Canada, add $3.00 per disk for shipping and handling. All payments must be in US funds.

□ Payment enclosed
Charge □ Visa □ MasterCard □ American Express

Acct. No. _____ Exp. Date _____
(Required)

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 4-5 weeks for delivery.

455113B

COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **800-334-0868** (in NC 919-275-9809) or write COMPUTE! Books, P.O. Box 5058, Greensboro, NC 27403.

Quantity	Title	Price*	Total
_____	COMPUTE!'s Commodore Collection, Volume 1 (55-8)	**$12.95**	_____
_____	Commodore Peripherals: A User's Guide (56-6)	**$ 9.95**	_____
_____	Creating Arcade Games on the Commodore 64 (36-1)	**$14.95**	_____
_____	Machine Language Routines for the Commodore 64 (48-5)	**$14.95**	_____
_____	Mapping the Commodore 64 (23-X)	**$14.95**	_____
_____	All About the Commodore 64, Volume I (40-X)	**$12.95**	_____
_____	COMPUTE!'s Commodore Collection, Volume II (70-1)	**$12.95**	_____
_____	COMPUTE!'s First Book of VIC (07-8)	**$12.95**	_____
_____	COMPUTE!'s Second Book of VIC (16-7)	**$12.95**	_____
_____	COMPUTE!'s Third Book of VIC (43-4)	**$12.95**	_____
_____	COMPUTE!'s First Book of VIC Games (13-2)	**$12.95**	_____
_____	COMPUTE!'s Second Book of VIC Games (57-4)	**$12.95**	_____
_____	Creating Arcade Games on the VIC (25-6)	**$12.95**	_____
_____	Programming the VIC (52-3)	**$24.95**	_____
_____	VIC Games for Kids (35-3)	**$12.95**	_____
_____	Mapping the VIC (24-8)	**$14.95**	_____
_____	The VIC and 64 Tool Kit: BASIC (32-9)	**$16.95**	_____
_____	Machine Language for Beginners (11-6)	**$14.95**	_____
_____	The Second Book of Machine Language (53-1)	**$14.95**	_____
_____	Computing Together: A Parents & Teachers Guide to Computing with Young Children (51-5)	**$12.95**	_____
_____	BASIC Programs for Small Computers (38-8)	**$12.95**	_____

*Add $2.00 per book for shipping and handling.
Outside US add $5.00 air mail or $2.00 surface mail.

Shipping & handling: $2.00/book _____

Total payment _____

All orders must be prepaid (check, charge, or money order).
All payments must be in US funds.
NC residents add 4.5% sales tax.
☐ Payment enclosed.
Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No._____ Exp. Date_____

Name_____

Address_____

City_____ State _____ Zip_____

*Allow 4–5 weeks for delivery.
Prices and availability subject to change.
Current catalog available upon request.

4559498

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!'s Gazette** for Commodore.

For Fastest Service
Call Our **Toll-Free** US Order Line
800-334-0868
In NC call 919-275-9809

COMPUTE!'s Gazette

P.O. Box 5058
Greensboro, NC 27403

My computer is:
☐ Commodore 64 ☐ VIC-20 ☐ Other_____

☐ $24 One Year US Subscription
☐ $45 Two Year US Subscription
☐ $65 Three Year US Subscription

Subscription rates outside the US:

☐ $30 Canada
☐ $65 Air Mail Delivery
☐ $30 International Surface Mail

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Payment must be in US funds drawn on a US bank, international money order, or charge card. Your subscription will begin with the next available issue. Please allow 4–6 weeks for delivery of first issue. Subscription prices subject to change at any time.

☐ Payment Enclosed ☐ Visa
☐ MasterCard ☐ American Express

Acct. No. _____ Expires ___/___
(Required)

The *COMPUTE!'s Gazette* subscriber list is made available to carefully screened organizations with a product or service which may be of interest to our readers. If you prefer not to receive such mailings, please check this box ☐.

755199

Writing Made Easy

Thousands of people have already made *SpeedScript* COMPUTE! Publications's most popular program ever. Offering nearly every feature and convenience you expect to find in a quality word processor, *SpeedScript* is the perfect writing tool. With *SpeedScript*, writing, editing, formatting, and printing any document—from the shortest letter to the longest novel—become easier. Anything can be changed, modified, or rewritten with just a few keystrokes on your Commodore 64 or VIC-20. The mechanics of writing become less intrusive—so you can concentrate on the *writing*, not the *process* itself.

SpeedScript 3.1 is our most powerful version of this easy-to-use word processor. Commands have been added, other features enhanced, to give you the best possible writing instrument.

Here are just a few of the things included in *SpeedScript: The Word Processor for the Commodore 64 and VIC-20*:

- Complete program listings for both the 64 and VIC-20 versions of *SpeedScript 3.1*
- Detailed documentation that shows you how to use all of *SpeedScript*'s commands and capabilities
- "The Machine Language Editor: MLX," an entry program which insures that you'll type in *SpeedScript* right the first time
- *SpeedScript 3.1* File Converter, a utility that lets you convert other files, such as records from a database, into *SpeedScript* files or allows you to turn *SpeedScript* files into true ASCII so that they can be sent to other computers via modem
- "ScriptSave," an automatic SAVE program you can use with *SpeedScript*
- *SpeedScript 3.1*'s source code (By studying it, you'll see how *SpeedScript* was written.)
- Clip-out quick-reference cards and keyboard overlay

SpeedScript is undoubtedly the best buy in word processors available today. It's a writer's tool that you can use from the moment you run it. With *SpeedScript: The Word Processor for the Commodore 64 and VIC-20*, you have a complete package: word processor, documentation, and utilities.

For the Commodore 64 and the VIC-20 (8K RAM expansion required).

ISBN 0-942386-94-9