MINCE
MARK OF THE UNICORN
222 Third Street
Cambridge, MA 02142
(617) 576-2760
For 520ST $00.00

by Tom Hudson

In my analysis of the 520ST computer system, **Atari's 520ST: Our First Look**, in issue 32 of **ANALOG Computing**, I stated that when we first received the 520ST, it had no text editor included for editing programs.

I neglected to mention that, within two days of receiving the computer, a package arrived from Mark of the Unicorn, Inc., a Cambridge, Massachusetts software company. In the package: you guessed it, the 520ST's first text editor, **Mince**. With **Mince** disk in hand, I proceeded to give it a good breaking in.

What is Mince?

Mince is an extremely powerful text editor for the 520ST computer. It won't work on any of the Atari 8-bit machines, so don't get a copy unless you have (or will have) an ST.

Notice that I said **Mince** is a *text editor*. It's not a full-blown word processor, although with some work and additional software to drive a printer, it could be used as the text editor portion of a word processor. I prefer to think of it more as an editor for creating and changing the source code of programs intended for the ST. In this application, the program does a fine job.

When you purchase the **Mince** text editor, you receive a 3½-inch disk and a large user's guide.

The disk, which you should copy for a backup as soon as you receive it, contains several important files: a configuration program, **Mince** itself, a "swap" file and three tutorial text files.

The configuration program is used to tell **Mince** all about the computer system you're using. **Mince** is designed to work on a number of computer systems, and must be told how to use the features of the 520ST.

To configure **Mince** for the ST, you execute the configuration program and select items from a menu to complete the process. With the version of **Mince** we received, I had a little trouble in the configuration for the 520ST.

The ST is listed as a standard computer on the "terminal type" menu, but a "keyboard bias" figure was incorrect in the standard terminal configuration. I modified these settings to their proper values, and the configuration was complete.

Using Mince.

After you've told **Mince** that it's working on an Atari 520ST, you're ready to start editing text. If you're familiar with the use of advanced text editors, you can probably go right into the **Mince** program itself and type away.

Some users will, of course, be new to the world of electronic text editing, so Mark of the Unicorn has thoughtfully provided eight lessons on the use of **Mince** in their manual. Three of these lessons (numbers four, six and eight) are included on the **Mince** disk, in order to familiarize new users with disk I/O operations.

Lesson one, "Getting Started," shows you how to load **Mince** from the disk. Interestingly, this process is slightly different from the manual's procedure in using the ST's GEM Desktop. You simply point the mouse at the icon labeled MINCE.PRG and click the button twice. **Mince** is then brought into operation.

Lesson one continues, showing how to move the cursor around on the screen and how to enter and modify text.

Lesson number two, "Getting Around Faster," teaches some of **Mince**'s more powerful cursor movement commands: move to beginning of line, end of line, forward a word, backward a word, and handy "universal" repeat function.

With the latter, you can specify repeat values for just about any keypress with a simple CTRL-U sequence. For example, the sequence *CTRL-U 10** will place ten asterisks at the current cursor position; *CTRL-U 10 [DELETE]* will delete the previous ten characters. **Mince** is simply packed with commands such as this, which make text manipulation fast and easy.

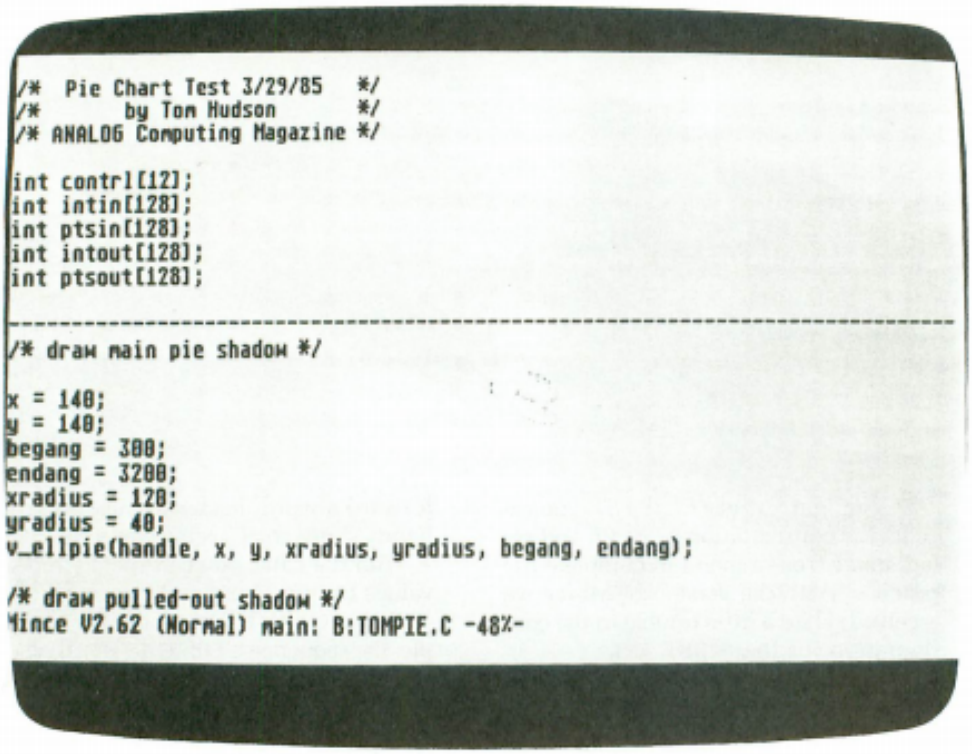
Lesson three, "Reading and Writing Files," teaches the fundamentals of working with a disk filing system, and how **Mince** uses disk files to store text. This lesson also explains the function of the "swap" file provided on the **Mince** disk.

It's used to hold older sections of text that you're not currently using. If you leave the keyboard idle for a specified time (set up when you configure **Mince**), the computer will take the opportunity to write modified pages to the swap file, so that any later page swapping can be done faster.

I have found this swapping action to be a minor nuisance when editing text. For example, pausing for a moment to think of the proper code to use in a program statement sometimes gives **Mince** the chance to do a "swap" operation.

If the swap takes a few seconds and is in progress when I'm ready to type again, I have to wait until the swap is complete before I can continue typing. As I said, it's only a minor complaint and doesn't happen when you're typing in a steady stream.

Lesson four, "Searching," which is included as a text file on the **Mince** disk, teaches about text searches in the program. **Mince** has the ability to search both forward and backward in a text file, which is a terrifically handy feature.



```
/*  Pie Chart Test 3/29/85   */
/*       by Tom Hudson       */
/* ANALOG Computing Magazine */

int contrl[12];
int intin[128];
int ptsin[128];
int intout[128];
int ptsout[128];

-------------------------------------------------------------------------------

/* draw main pie shadow */

x = 140;
y = 140;
begang = 300;
endang = 3200;
xradius = 120;
yradius = 40;
v_ellpie(handle, x, y, xradius, yradius, begang, endang);

/* draw pulled-out shadow */
Mince V2.62 (Normal) main: B:TOMPIE.C -48%-
```

A Mince screen, showing the use of multiple windows.

Mince can also default to the latest search used, if you like. For example, if you search for the word *antidisestablishmentarianism*, and **Mince** finds it, you can stop, perform other editing funtions, then search for *antidisestablishmentarianism* again with only two keystrokes. This can obviously save a lot of typing headaches!

Lesson five, "Keyboard Culture," is a short section discussing conventions used in the world of computers and the subsequent **Mince** lessons, regarding special control-key sequences.

Because it has so many commands available for you, **Mince** must use two types of command key sequences: control characters and meta-commands. Got that?

Control (CTRL) characters are simply normal keyboard keys struck with the CTRL key pressed at the same time. CTRL-S, for example, starts a forward string search, and CTRL-K kills (erases) a line of text.

Meta-commands are prefixed with the ESC key. ESC-D deletes the next word in the text buffer; ESC-U makes the word under the cursor all uppercase characters, and so on.

Some commands use a combination of meta-commands and CTRL characters, such as ESC-CTRL-R, which performs a query on a search-and-replace operation. As you can see, with over 86 text manipulation commands, **Mince** must rely on several key sequences to access all the available commands. Fortunately, Mark of the Unicorn provides a welcome command summary card, which I have hanging in a convenient location above the 520ST.

Lesson six, called "Killing and Moving Text," gives a complete wrap-up of the various commands which allow you to delete or move characters, lines or whole blocks of text.

Deleting a block is easy; you simply place a text mark at the beginning of the text, move your cursor to the end of the block to delete and press CTRL-W, or "Wipe Region." The text instantly disappears.

What if you make a mistake? **Mince** allows you to re-insert the text you deleted by pressing CTRL-Y, or "Yank back deleted text." CTRL-Y can be used any number of times, to place the text in as many places in the text buffer as you like. By using the CTRL-W and CTRL-Y commands together, you can move or copy a block of text anywhere in your document with only a couple of keystrokes.

Fortunately, **Mince** will allow you to save the text from several block delete operations and place the new, larger block elsewhere in the text.

Lesson seven, "Text Processing Commands," gives information on processing words, sentences and paragraphs. **Mince** has a powerful set of commands which enable you to capitalize (change the first character to uppercase), uppercase or lowercase the entire word, starting with the current character. These commands can really be timesavers.

Other commands allow deletion of sentences (ending with periods), right-justification and indentation of paragraphs, and so on.

Finally, lesson eight covers the use of text buffers, a powerful feature which allows **Mince** to store several documents in the computer's memory at the same time. Not only can you look at any of the buffers while they're in memory, but you can use the text movement commands to move text from one buffer to another.

There's even more.

The eight lessons in the **Mince** manual are intended to get the new **Mince** user accustomed to ordinary text entry and manipulation operations. I've tried to give you a fairly complete summary of **Mince**'s commands, although there are far too many to completely summarize here.

Advanced **Mince** users, or those who have used a computer text editor before, can read more about the editor in the *Mince User's Guide*. This guide contains information on creating multiple text windows, where the screen is divided into two sections, each showing a different portion of the text document.

After using **Mince** for more than two months, I can say that I've found it an easy-to-use, well-documented program. With its eight tutorial-style lessons, it can allow even the computer novice to use it effectively, as well as the seasoned programming professional.

So far, all documents prepared with **Mince** have performed perfectly in conjunction with the compiler and assembly programs **ANALOG Computing** uses on the 520ST.

I'm not going to mince words—Mark of the Unicorn's **Mince** is an excellent first editor for the 520ST. And, if it's any indication of the software to come for Atari's 16-bit machine, we've got a lot of good programs to look forward to. □