

T/Maker™

T/MAKER

Registration Card

I have read the license agreement for T/Maker and agree to all terms and conditions. I hereby request to be registered as an authorized user.

Please PRINT

Serial Number _____ Version _____

Purchased from

Your Name

Company

Address

City _____ State _____ Zip _____

Signature

Date

BUSINESS REPLY MAIL

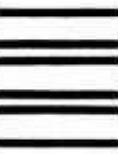
FIRST CLASS

PERMIT NO. 100

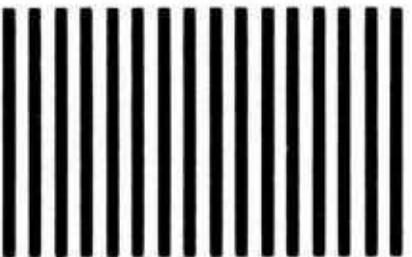
MTN. VIEW, CA

Postage will be paid by

*T/Maker Company
2115 Landings Drive
Mountain View, CA 94043*



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES





Copyright 1985 T/Maker Company

All Rights Reserved

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of T/Maker Company.

Disclaimer

No representations or warranties are made with respect to the contents hereof. Further, T/Maker Company reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation to notify any person of such revision.

T/Maker Company
2115 Landings Drive
Mt. View, California 94043

SOFTWARE LICENSE

T/Maker Company agrees to grant and the customer agrees to accept a nontransferable and nonexclusive license to use the T/Maker Integrated Software programs delivered with this agreement on the basis of the following terms and conditions. The licensed programs are sold for use on a single computer system. For computer systems, capable of serving more than one user simultaneously, site licenses are required and available from T/Maker Company.

The customer may make copies of the licensed programs for backup purposes or for use by the customer on the applicable computer system. The documentation, however, may not be reproduced under any circumstances.

Welcome to the world of T/Maker Integrated Software!

WHAT'S IN YOUR T/MAKER PACKAGE?

- * A Reference Manual with a Tutorial
- * Quick Reference Booklet(s) [1]
- * T/Maker Master Diskette(s)

STARTING UP

Some T/Makers are "preconfigured" (that is, ready to run). Other T/Makers are not, and must be "initialized." The T/Maker ADDENDUM included in your package will tell you whether your T/Maker is preconfigured or not.

If your T/Maker is preconfigured:

Start T/Maker by typing:

A> TMAKER

If your T/Maker is not preconfigured:

You must "initialize" your T/Maker system. Turn to the "Customization" section of this manual for instructions.

USING T/MAKER

Once you have typed TMAKER, the T/Maker copyright and the WHAT NEXT?-prompt appear on your screen.

You are now ready to go! We suggest that you take the following steps:

- 1) Read the T/Maker INTRODUCTION section of the Reference Manual.
- 2) Go through the T/Maker TUTORIAL.

Once you have digested the INTRODUCTION and the TUTORIAL, you will be ready undertake your own T/Maker tasks.

Have Fun!

[1] Not included under some circumstances.

TABLE OF CONTENTS

THE INTRODUCTION	SECTION ONE (IN)
A TUTORIAL	SECTION TWO (TT)
FILE MANAGEMENT	SECTION THREE (FM)
WORD PROCESSING	SECTION FOUR (WP)
SPELLING CHECKER	SECTION FIVE (SC)
DATABASE FUNCTIONS	SECTION SIX (DB)
SPREADSHEET	SECTION SEVEN (ES)
BAR CHARTS	SECTION EIGHT (GR)
LIST PROCESSING	SECTION NINE (LP)
TRANSFER UTILITIES	SECTION TEN (TU)
DATA ENCRYPTION	SECTION ELEVEN (DE)
DESIGNING SYSTEMS.	SECTION TWELVE (DS)
CUSTOMIZATION.	SECTION THIRTEEN (CN)
INDEX.	SECTION FOURTEEN (IX)
QUICK REFERENCE PAGES	SECTION FIFTEEN (QR)

THE INTRODUCTION

THE STRUCTURE OF T/MAKER

INTEGRATED SOFTWARE

What Can T/Maker Do?

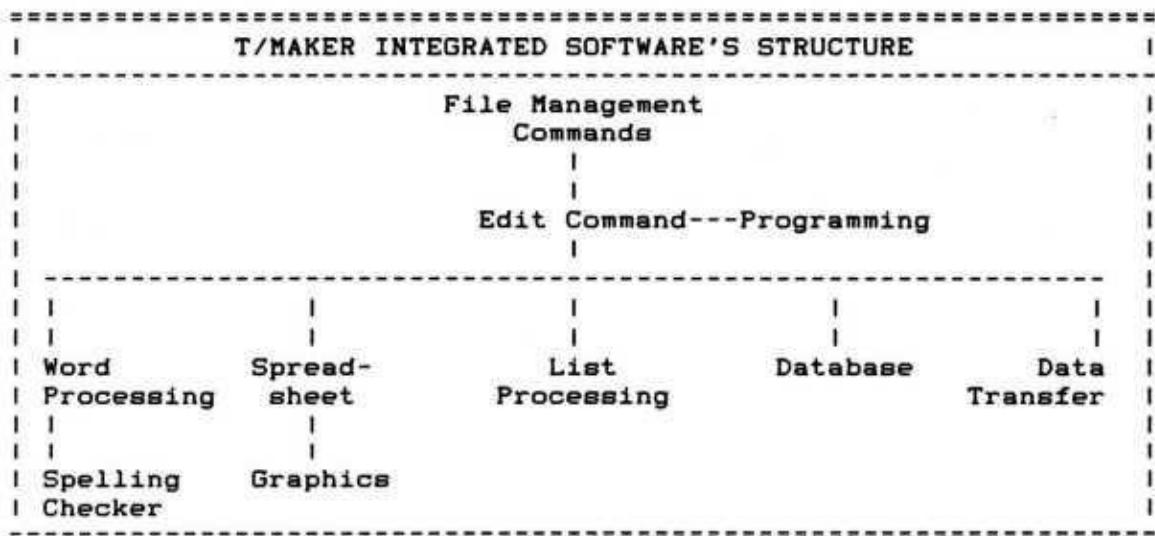
"T/Maker Integrated Software" is T/Maker Company's latest and most powerful integrated software package. "Integrated software" refers, of course, to single software packages that do several sorts of things. T/Maker does nine:

- File Management
- Word Processing
- Spell Checking
- Spreadsheet
- Database Management
- List Processing
- Data Transfer
- Graphics (Bar Charts)
- Programming

T/Maker also provides (1) on-the-screen HELP and (2) a tutorial for teaching the program to yourself.

Overview of T/Maker

Below is a chart of T/Maker's basic organization.



This structure encompasses some fifty different T/Maker Commands.

T/Maker commands are keyed-in after T/Maker's "WHAT NEXT?"-prompt. This prompt appears at the bottom of your computer's screen when T/Maker is running.

All T/Maker work is done on "files." A file is simply a single entity or package of information: a business letter might be a file, a table of numbers might be a file, so too might be a database, a report, or the chapters of a book.

Your computer can both do work on files and store files for later use. To do work on a file that file must be on your computer's screen--the screen is also called your computer's "memory." The file on the screen is called the "working file." You store files on disk--either on floppy disks or on a hard disk. Files on disk are called "disk files."

File Management Commands

"Managing" files means doing such things as creating a file (with its own unique name), saving a file on disk, retrieving a file from disk and placing it on the screen, deleting a file from disk.

In all, T/Maker provides seventeen file management commands:

```
=====
          T/MAKER'S FILE MANAGEMENT COMMANDS
=====

CREATE      creates a working file with a unique name.
SAVE        stores the working file on disk.
DELETE      discards a file on disk.
FILES       screens a directory of files on disk.
GET         retrieves a file from disk, making it the working
            file.
RENAME      lets you give the working file a new name.
DATA        sets the default disk drive.
INSERT      lets you add the contents a disk file to the
            working file.
MERGE      lets you overlay the contents of a disk file
            on the working file.
1/1         lets you send the cursor to the desired line and
            column position in the file.
CLIP        discards portions of the working file.
LIST        lets you review the contents of a disk file
            without displacing the current working file.
INFO        temporarily displays a variety of facts about the
            working file.
TABS        lets you save tab settings with a file.
NOTABS     lets you discard tab settings when file is saved.
WAIT        lets you interrupt a series of T/Maker commands.
STOP        terminates T/Maker.
=====
```

These are simple commands to use. Most often, they involve merely a single command word or a single command word and the name of the file you wish to execute the command upon. No matter what sort of task you wish to accomplish with T/Maker--word processing, spreadsheet calculations, database management, list processing, or data transfer--you will be working on files and thus you will need to use file management commands.

The EDIT Command

As mentioned, a file you wish to work upon must be on the computer's screen. You put a file on the screen either by creating it or retrieving it from disk. Once the file appears on the screen, you must "get into the file" itself to work on it. You do this by using T/Maker's EDIT Command. No matter what the task, the EDIT Command puts you "into the file" letting you do work upon it. You invoke the EDIT Command by keying in "EDIT" or "E" (for short) after the "WHAT NEXT?"-prompt. This is called "entering the editor."

Automating T/Maker Command Sequences

T/Maker commands do not have to be given one at a time. They can be keyed-in in a series after the "WHAT NEXT?"-prompt. T/Maker will then carry out such series in order. T/Maker can store such sequences, too. T/Maker's DO Command provides the device for invoking stored series of commands.

Word Processing

Three T/Maker commands provide for word processing:

===== T/MAKER'S WORD PROCESSING COMMANDS =====

EDIT (as above) lets you enter or revise text.
ALIGN lets you set margins, dictate the text's
 justification, and center text.
PRINT prints text.

T/Maker also provides an array of "Design Commands," which are incorporated into the file itself. These allow you to shape the appearance of finished documents in a great variety of ways (for example, they let you single or double-space text, insert footnotes, provide automatic page numbering, design footers and headers for pages, etc.).

Spell Checking

Four T/Maker commands concern spelling:

===== T/MAKER'S SPELLING COMMANDS =====

PROOF checks text for spelling errors, typos, and
 unfamiliar words.
CORRECT allows you to correct errors.
AMEND lets you add or delete words from your dictionary.
SPELL provides help in finding a word's correct
 spelling.

Spreadsheet

T/Maker approaches numerical problems visually. There are two steps to this approach: First, you lay out the calculation you wish done on your screen (once more, use the EDIT Command for this); second, you invoke the COMPUTE Command. T/Maker's COMPUTE Command carries out the calculation and fills in results. Two

more commands round out T/Maker's spreadsheet functions. COMBINE lets you carry out calculations simultaneously involving the working file and a disk file, and CLEAN removes calculation symbols from a finished table.

Graphics

Properly formatted calculations can easily be rendered as bar charts using T/Maker's BAR Command. A variety of options allow you to shape the bar chart's appearance.

List Processing

Eight commands process list data, though some of these commands also have applications to non-list sorts of data or text:

```
=====
          T/MAKER'S LIST PROCESSING COMMANDS
=====
SORT      sorts lists into alphabetical or numerical order,
either in ascending or descending direction.
TALLY    counts like elements in lists and also can sum up
numerical information associated with such counts.
KEEP/DROP prune lists (or other sorts of text) of lines
containing specified strings of characters.
FIND      searches out specified strings.
REPLACE   replaces a specified string with another.
ARRANGE   moves a list's columns to new locations.
MATCH     contrasts the contents of two files.
=====
```

Database Functions

A database is the computer equivalent of a box of index cards, each card holding a standardized collection of information. You establish a database file by showing T/Maker what a standard card (called a "record") should look like. This is done with the Editor. Once a record is described, you may use a variety of database commands to manipulate the file:

```
=====
          T/MAKER'S DATABASE COMMANDS
=====
UPDATE      allows you to enter, revise, or delete data.
COMPLETE    transfers data in bulk from one file to another.
ORDER       puts records into alphabetical or numerical order.
GROUP       makes summary reports from a database file.
SELECT      allows you both to select particular records and
            reformat their appearance.
SET         allows you to make mathematical computations and
            perform logical tests on data in records.
=====
```

Data Transfer

Data transfer commands permit you to move data in an ad hoc fashion from one file to another without the necessary structure associated with a database:

```
=====
          T/MAKER'S DATA TRANSFER COMMANDS
=====
LOAD        draws data from a disk file, filling them into the
            working file.
UNLOAD      transforms the working file into a format suitable
            for supplying data to other files.
TRANSFER    moves a datum from file to file.
=====
```

Below is a summary of the T/Maker commands that make up each of T/Maker's eight utilities.

INVENTORY OF T/MAKER COMMANDS (ALPHABETICAL)		
The EDIT Command	WORD PROCESSING:	LIST PROCESSING:
FILE MANAGEMENT:	ALIGN EDIT PRINT	ARRANGE DROP FIND
CLIP		MATCH
CREATE		REPLACE
DATA	SPELL CHECKING:	SORT
DELETE		TALLY
FILES	AMEND CORRECT PROOF SPELL	DATABASE:
GET		COMPLETE
INFO		GROUP
INSERT		ORDER
LIST		SELECT
MERGE	DATA TRANSFER:	SET
NOTABS	LOAD TRANSFER UNLOAD	UPDATE
RENAME		
SAVE		
STOP		
TABS		
WAIT	PROGRAMMING:	GRAPHICS:
1/1	DO HELP	BAR

How Does T/Maker Do Things?

The Intelligent Chalkboard

T/Maker takes a visual approach to tasks. Its approach might be likened to that of a magical, intelligent chalkboard.

Imagine for a moment that you possessed a chalkboard that did what ordinary chalkboards do--that is, it provided a writing surface--but also had the power to do various kinds of work on the things you wrote upon it. Suppose, for example, that your chalkboard could actually solve the mathematical problems you wrote upon it, or correct the spelling of words you wrote on it, or organize into alphabetical order a list you wrote on it. T/Maker provides just these sorts of powers to your computer.

T/Maker turns your computer--more specifically, your computer's working file--into that intelligent chalkboard. When you are using T/Maker you are always in one of two modes. You can

be either (1) standing right next to the chalkboard, writing things upon it, or (2) standing back from the chalkboard, waiting for T/Maker to do the task you have asked it to do.

- (1) "Outside the Editor": When you are standing back from the chalkboard, T/Maker's screen presents a WHAT NEXT?-prompt at its lower left corner. That prompt is T/Maker's way to invite you to use one of T/Maker's many commands.

Walking away from the chalkboard is called "leaving" or "quitting the editor." You quit the editor by pressing the quit-the-editor keystroke (this keystroke varies according to the kind of computer you have).

- (2) "Inside the Editor": When you are standing right next to the screen, the WHAT NEXT?-prompt disappears from the screen. Now, your cursor lies directly upon the blank surface of the chalkboard. You can write on the chalkboard using your keyboard like a typewriter.

In T/Maker lingo, going up next to the chalkboard is called "entering the editor." You enter the editor by typing EDIT or E (for short) in response to the WHAT NEXT?-prompt and pressing the RETURN key.

Using the Chalkboard

The working file can be thought of as a big piece of paper which is 300 column-positions wide (400 for DOS). The total size of the working file is determined by the memory of your computer. A typical CP/M machine with 64 kilobytes of memory will provide about 23,000 characters of memory (45,000 characters for MSDOS machines with 128K bytes). How big a file is that? If you divide this number by the average length of a line in your file, you will get a rough limit for the number of lines you may have in the file.

Obviously, the working file is much bigger than the screen of your computer. You should think of your computer's screen as a window for looking at the working file. This window can be moved up and down or sideways to look at all parts of the file.

The presence of T/Maker's WHAT NEXT?-prompt at the lower left of your screen means that T/Maker is waiting for a command. Any of some fifty commands can be given. For example, to get a file called "BUDGET.11" from disk and make it the working file, you type,

1/1 WHAT NEXT? GET BUDGET.11

Answers to the WHAT NEXT?-prompt are always followed by pressing the RETURN key. Answers can be typed in either upper or lower case letters. In the above command, "GET" can be thought of as the command-name and "BUDGET.11" as a "parameter" of the command. So, this command really consists of two parts. Some commands have no parameters and only the command-name, itself, is required to use them. Other commands may need various parameters or options to explain what exactly is to be done.

Though file management commands tend to deal with the working file as a whole (for example, "GET" gets a whole file from disk, "SAVE" saves the whole file on disk, and "DELETE" deletes the whole file from disk), many T/Maker commands affect only a part of the working file. This area of influence is called the "working area" or the "active area."

The active area is determined by the cursor's location in the file. There are two ways to define the active area:

- (1) When you are inside the editor, the active area runs from the location of the cursor to the end of the file. In other words, everything above the cursor is not in the active area.
- (2) When you are outside the editor (and the WHAT NEXT?-prompt is on your screen), the active area runs from the top line on the screen to the end of the file. Again, everything above the screen's top line is not in the active area.

For example, the SORT command, which sorts lines into alphabetical or numerical order, sorts only lines in the active area. Any lines above the screen's top (when outside the editor) will not be sorted when the SORT command is used.

A few T/Maker commands deal with only one line in a file. The DO command, for example, extracts a single line from the file and interprets it as an answer to the WHAT NEXT?-prompt. To describe such commands, the expressions "Top Line on the Screen" and "Top Line in the Frame" are frequently used.

The diagram on the last page of this section summarizes the terms introduced so far.

A final group of commands are not answers to the WHAT NEXT?-prompt at all. These commands take the form of lines of text placed in the working file.

For example, to align text (or justify or center text), you must put information in the working file that tells what text is to be justified, how it is to be justified, and where the margins are located. All this is done by placing "aligning wedges" in a file on a blank line like this:

<<

>>

Such commands typically stay in the file as a regular part of it and do not print when the file itself is printed out. Because of their static nature, you might prefer to think of them as specifications and not commands at all, since typing them into a file does not bring about an immediate action. Only when an action command is given as the response to the WHAT NEXT?-prompt are these specifications interpreted and enforced.

WORKING FILE

(All Lines)

□ Line 1 Position 1 of the Working File

SCREEN (the "window")

Working

or

Active

Area

.

.

.

.

.

.

.

.

.

.

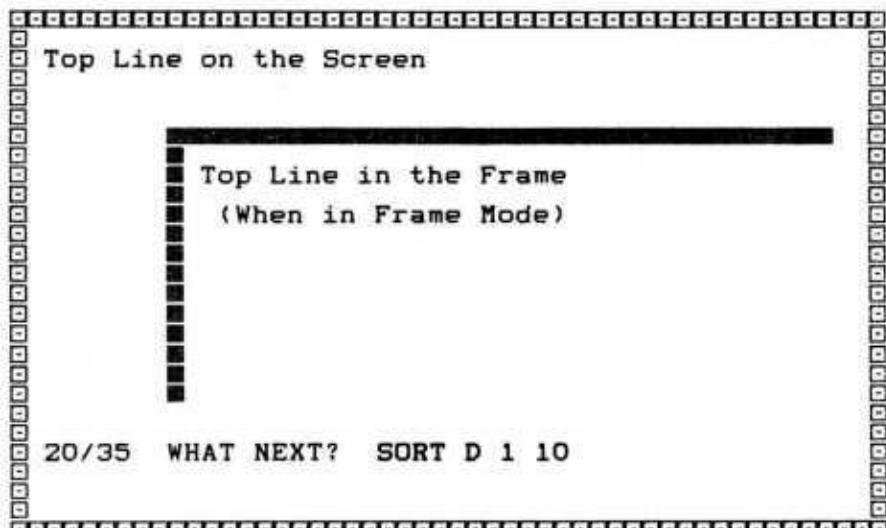
.

.

.

.

V



Command Options
or Parameters

Command-name

Universal Prompt

Line and Position Coordinates of the Upper
Left-Hand Corner of the Screen (or of the
Frame, when in Frame Mode).

GETTING STARTED WITH T/MAKER

a 45-Minute Tutorial

Welcome to T/Maker. This tutorial should familiarize you with T/Maker so you can become productive with it in a short period of time. To do this, we've set up typical business examples for you to experiment with. Because they are real T/Maker files, there is no reason why you couldn't use them in your everyday work as well. Here's the agenda:

WHAT YOU WILL LEARN:

Topic	Activity
Starting T/Maker	Getting your program going
Word Processing	Writing a Letter
Data Base Management	Storing Names & Addresses
Integration	Creating a Form Letter
Spreadsheet	Creating an Invoice
Peek At Other Capabilities	"What If" Analysis
	Bar Charts, Spell Checking

The best approach is to find a free, relaxed 45-minute period (*probably the hardest instruction in this tutorial!*) and complete all the activities in one sitting. The next page lists some special instructions to make the job easier. Good luck, and enjoy!

Things You Should Know:

There are two versions of T/Maker; one for PC-DOS and compatibles (which uses mostly function keys for editing) and one for other machines (which does not use function keys.) If you have a PC or compatible, use the keystrokes in the squares -- they are the keystrokes which always appear on the left. If you have any other computer, use the keystrokes in the rounded boxes -- they always appear on the right.

Another word on keystrokes: When it is indicated to use the ESCAPE key with another key, you hit the ESCAPE key *first*, then hit the other key. (TRS80 Model 4/4P users -- your ESCAPE key is labeled BREAK.) If it is indicated to use the CONTROL key (CTRL) with another key, you hold the CONTROL key down while you hit the other key *simultaneously*. Therefore, the sequence:

would be accomplished by first hitting the ESCAPE KEY (and then letting it go) then holding down the CTRL key while hitting the O key at the same time.

Anything we want you to type is in **bold** characters.

If you get the error message "COMMAND _____ FAILED" you have either mistyped the name of the command, or the command you need is not on the disk you have in the drive. Try to type it again, and if you still get the message, replace the disk in the drive with another of the T/Maker disks you received and try the command again.

If the command you type in does not seem to work, type a **I** at the WHAT NEXT? prompt, and try the command again. As you move through the tutorial, you'll find an explanation of what this does.

If you hopelessly mess up the file you are working on, just turn the tutorial back to the last time you used a GET command, and start over from there, beginning with the GET command itself.

Starting T/Maker

Turn on your computer and load the operating system. After getting to the operating system prompt, place the T/Maker Start Disk in drive A and type:

A>TMAKER

In moments, you should see the T/Maker Starting Screen with the T/Maker copyright notice, and the WHAT NEXT? prompt at the bottom of the screen.

Of Chalkboards and Computers: Writing Our First Letter

Wouldn't it be nice if a computer could be used as easily as a chalkboard or a fresh sheet of paper?

Think of T/Maker as an *electronic* chalkboard. Right now we are standing away from the chalkboard. We know this because we can see the WHAT NEXT? prompt at the bottom of the screen.

T/Maker is wondering what to do next. We can tell T/Maker what to do by typing in *commands* at the WHAT NEXT? prompt.

Let's start our first example. Suppose we want to get a letter that we had started before but couldn't finish. Let's get the letter and finish it now.

By using the T/Maker command GET, T/Maker will get and present to us the chalkboard named LETTER. Type in the following command at the WHAT NEXT? prompt:

1/1 WHAT NEXT? GET LETTER (then hit the return key)

Something should appear on the screen that looks like a letter. How can we check and see if it's our unfinished letter?

To find out information about the current chalkboard, we'll use the command INFO. Simply type INFO at the WHAT NEXT? prompt.

1/1 WHAT NEXT? INFO (*return*)

The *info line* overlayed at the top of the screen tells us that we do indeed have the LETTER chalkboard.

Notice for a second the two numbers to the left of the WHAT NEXT? prompt. They represent where we are on the T/Maker chalkboard (row/column).

Because the chalkboard is so large, we can only view a small piece at a time. By typing a coordinate at the prompt, we can change our view of the chalkboard. Let's try it. Type 5 at the WHAT NEXT? prompt:

1/1 WHAT NEXT? 5 (*return*)

The view of the chalkboard slips down so that the 5th row now becomes the top row of the screen. Notice that the 1/1 in front of WHAT NEXT? has changed to a 5/1, indicating that we are now on line 5. Don't worry, we haven't lost the first 4 lines! They are above the top of the screen in the *inactive* area of T/Maker's chalkboard.

Type 1 to move our view back to the top of the chalkboard:

5/1 WHAT NEXT? 1 (*return*)

See, our top 4 lines reappear, and now the entire letter is once again in the active area.

If you mistype a command, or ask T/Maker to do a command that is not contained on the current T/Maker command disk, T/Maker will tell you that it can't find the incorrect or unavailable command.

Let's try typing an incorrect command and see what happens. Type UNFO at the WHAT NEXT? prompt:

1/1 WHAT NEXT? **UNFO** (*return*)

This *error message* should appear at the bottom of your screen:

COMMAND UNFO FAILED: NONEXISTENT
HIT A KEY TO CONTINUE

To recover from a mistake, simply press any key. The WHAT NEXT? prompt should reappear at the bottom of the screen.

If you spelled the T/Maker command correctly and still get this message, the command probably resides on another T/Maker disk. Simply swap disks and retype the desired command.

Writing on the Chalkboard: The EDIT Command

We tell T/Maker what to do by typing commands at the WHAT NEXT? prompt. We said that this was like standing *away* from the chalkboard.

We can move up to the chalkboard to write on it by using the command EDIT. Type EDIT or E at the WHAT NEXT? prompt:

1/1 WHAT NEXT? EDIT (*return*)

We know that we've moved up to the chalkboard because the WHAT NEXT? prompt has disappeared. Notice the *cursor* (sometimes a blinking character) in the upper left-hand corner of our board. We've stepped up to the chalkboard and could now begin writing on it.

We can easily step away from the chalkboard again by giving the keystroke to exit the editor. Press the key

F1

(*Exit the Editor*)

CTRL-Q

Notice that the WHAT NEXT? prompt reappears at the bottom of the screen, and the cursor is placed immediately after it. We are once again standing away from the chalkboard.

Moving Around: Cursor and Editing Keystrokes

The EDIT command lets you write on the T/Maker chalkboard. Let's step up to the chalkboard again by typing EDIT at the WHAT NEXT? prompt, so that we may begin to write:

1/1 WHAT NEXT? EDIT (*return*)

Editing keystrokes give you full control of your chalkboard - that is, the ability to quickly move around the board and insert or manipulate text as needed.

The editing and cursor movement keystrokes used in T/Maker are listed Quick Reference pages in the back of the manual. You may want to glance at these keystrokes now.

Let's try moving our cursor around our letter chalkboard. Press the following cursor movement keystrokes:

PC Version	Non-PC Version	Cursor Action
→	→	to move cursor one space to right
↓	↓	to move cursor down one line <i>(repeat twice)</i>
END	CTRL-P	to Advance to the next word
ESC END	ESC CTRL-P	to Advance to the end of line
PG DN	ESC ↓	to move a page down
RETURN	ENTER	to move to beginning of next line
HOME	CTRL-O	to move to upper left-hand corner of screen (<i>home the cursor</i>)
ESC HOME	ESC CTRL-O	to move to 1/1 (<i>home the screen</i>)

Now let's try moving some text around. We should start from the top. If you haven't done so yet, press the keystrokes to home the screen.

ESC **HOME** (*Home the Screen*) **ESC** **CTRL-O**

Now try pressing the following keystrokes. (Hit each one a few times, and don't worry if you accidentally delete parts of this letter. You can always get a fresh copy of the original letter by exiting the editor and then asking T/Maker once again to **GET LETTER**.)

PC Version	Non-PC Version	Action
INS	CTRL-W	to insert a character
DEL	CTRL-E	to delete a character
F9	CTRL-R	to insert a line
F10	CTRL-T	to delete a line

Searching and Replacing Text

T/Maker includes all kinds of editing short-cuts. Suppose you would like to find or alter a certain word or phrase in a long document. T/Maker has special keystrokes that will do this automatically for you. Let's try finding the first occurrence of a word. Again, start from the top. Press the cursor keystroke to home the screen:

ESC **HOME** (*Home the Screen*) **ESC** **CTRL-O**

To find a word, press:

F5

(Search for String)

ESC

,

T/Maker will ask for the word to find. T/Maker is very precise -- it looks for the *exact* capitalization of the word you wish to find. For example, T/Maker thinks *ANIMAL* and *Animal* are two different creatures!

Why not search for *PLEFCO*?

1/1 FIND WHAT STRING? **PLEFCO** (*return*)

The command line disappears, and the cursor now rests on the first occurrence of PLEFCO. To find the next occurrence, press:

F6

(Continue Search)

ESC

&

Suppose we think that PLEFCO is dull name, and want to change it to BOBCO. We will use the REPLACE command. Press:

ESC**F5**

(Replace String)

ESC

"

T/Maker will ask you for the word to replace. With capitalization in mind, type PLEFCO.

9/1 REPLACE WHAT STRING? **PLEFCO** (*return*)

T/Maker will ask you what you want to replace it with. Answer by typing BOBCO.

9/1 WITH WHAT STRING? **BOBCO** (*return*)

Notice that every occurrence of PLEFCO *after the cursor* has been replaced with BOBCO. To have had them all replaced, you would have first homed the screen -- then the REPLACE function would have operated all the way from the top.

We can also get information about our chalkboard from within the editor. To do so, press:

F2

(*Info within Editor*)

ESC

(?)

To clean the *info line* off our screen, let's ask T/Maker to *redraw* the screen. Press:

ESC

F2

(*Refresh Screen*)

ESC

(#)

Inserting Text

Let's insert some new text into the letter. The simplest method of doing so is called *character overtype*. Just as it sounds, it will allow us to type right over the characters that are already there. To try this out, first get a fresh copy of the LETTER. Use your *Quit the Editor* keystroke to get back to the WHAT NEXT? prompt, and then type GET LETTER EDIT:

1/1 WHAT NEXT? GET LETTER EDIT (*return*)

(Notice a trick above. T/Maker will allow you to *stack* more than one command together before you hit the return key. We stacked both the GET command and the EDIT command together.)

Now move the cursor so that it rests on the first position of the line starting with *We are having* and type the following sentence *over* the existing sentence. (If you make a mistake, simply use your cursor movement keys to get back to the mistake and write over it. T/Maker is much neater to use than white-out!)

Type this sentence:

This is a one-time offer to our best

We can use our *Delete to End of Line* keystroke to erase the rest of the old first line. From the end of the new sentence (which is right where your cursor should be) press:

ESC **DEL** (*Erase to Line end*) **CTRL-Y**

Suppose we actually wanted to have added the words: *This is a one-time limited special offer to our best*. We could make room for the words we forgot by pushing the rest of the sentence over with our *Insert Character* keystroke. Move your cursor so that it rests between the words *time* and *offer*. Now press the following:

INS*(Insert Character)***CTRL-W***(press about a dozen times)*

Now you can type the words you forgot into the newly-created space.

Moving Blocks of Text

Suppose we decide that we want to have the first paragraph come later in the letter. To move sentences or paragraphs in T/Maker, you'll use the *buffer*.

A buffer is a temporary storage area. We'll tell T/Maker to *move* a full line of text into the buffer by placing our cursor on the line we want moved. Let's move to the top line of our document by pressing the *Home Screen* keystrokes.

ESC**HOME***(Home the Screen)***ESC****CTRL-O**

Use the *Down Cursor* key to place your cursor on the first sentence, then press:

F8*(Move Line to Buffer)***CTRL-G**

Notice that your line disappears. It is not gone, it has just been moved into a non-visible buffer. Continue pressing the *Move to Buffer* keystroke until the entire first paragraph resides in the buffer.

Next, using the *Down Cursor* key, move the cursor so that it rests on the first line of the current third paragraph, the paragraph we changed moments ago. We'll insert the buffer here by pressing:

ESC **F7**

(*Insert the Buffer*)

ESC **D**

The paragraph reappears in the new location. Extra blank lines can easily be added to separate paragraphs, if desired, by using the *Insert Line* keystroke.

Recording a Series of Keystrokes

Suppose we want to move the entire paragraph over 3 spaces. We can use T/Maker's *Keystroke Macro*.

A keystroke macro is like a tape recorder of keystrokes - that is, you can use it to record a series of keystrokes which you will then use over again later. Let's try it.

Move to the top row of the letter by pressing the *Home Screen* keystrokes.

ESC **HOME**

(*Home the Screen*)

ESC

CTRL-O

To begin a keystroke macro, press:

F3

(*Begin Keystroke Macro*)

ESC

CTRL-B

The recorder is now on and every keystroke you press will be recorded. Record the following keystrokes: Hit the *Insert Character* keystroke three times and the *Down Cursor* once. (If you have forgotten what they are, look at the keystroke tables you worked with earlier in the tutorial.) To end the recording of keystrokes, press:

F4

(*End Keystroke Macro*)

CTRL-X

These keystrokes are now stored and can be played back once or frequently at any time until you exit the editor.

To play back the keystroke macro recording, press:

F4

(*Execute Macro Once*)

CTRL-X

Notice that your text seems to jump to the right three places, with the cursor ending up at the beginning of the next line. Go ahead and continue to hit the *Execute Macro* keystroke until all the lines have been moved over. It's certainly easier than having to repeat those keystrokes yourself!

Justifying or Aligning Text

Suppose we want to straighten up our chalkboard. T/Maker uses *Aligning Wedges* to set margins. Representing left and right margins visually in this way makes setting margins easy and fast.

If you haven't done so already, let's exit the editor (F1 or CTRL-Q) and get another fresh copy of our original LETTER chalkboard. (You haven't learned to SAVE changes to a document yet, so the file LETTER on disk remains unaltered.) Type the following:

1/1 WHAT NEXT? GET LETTER (*return*)

The aligning wedges on the top line of this chalkboard (they look like "greater than" and "less than" mathematical signs) tell T/Maker the desired left and right margin.

To have T/Maker justify the letter to the desired margins, we'll use the ALIGN command, like this:

1/1 WHAT NEXT? ALIGN (*return*)

Look what happens! T/Maker straightens our letter in an instant. In this aligning case, we've asked for left and right justification. There are many other options.

It's easy to change the margins. Let's squeeze the them by four spaces.

First, we need to step up to the chalkboard.

1/1 WHAT NEXT? EDIT (*return*)

As usual, entering the editor causes the WHAT NEXT? prompt to disappear, and the cursor is again waiting for you at the top of the screen. Using your *Right Cursor* key, move four or five spaces to the right. Do not use the space bar, as this will wipe out your left aligning wedge. Now, press the *Delete Character* key four times:

DEL

(*Delete Character*)

CTRL-E

Then exit the editor. With the full letter now in T/Maker's *active area* (below the top line of our screen), you can give T/Maker the ALIGN command. Type ALIGN at the WHAT NEXT? prompt.

1/1 WHAT NEXT? **ALIGN** (*return*)

T/Maker re-aligns the letter to your new margin settings in a flash.

As mentioned above, you can specify various margin settings with a different combination of aligning wedges. Let's ask T/Maker for HELP.

T/Maker itself can give you most of the information you need to run it. For example, let's call up the HELP topic *Align* and take a look at what our options are. To do this, you simply type:

1/1 WHAT NEXT? **HELP ALIGN** (*return*)

You should now be looking at the HELP topic ALIGN. When you find the option you want, you can easily return to exactly where you were by ending the HELP session. Just type END at the HELP prompt.

1/1 Enter <topic> <HELP> <END> <?> **END** (*return*)

Where were we? Ah, yes. We just finished aligning the LETTER chalkboard.

Now that the letter looks like you want it to, you can save a copy permanently to the disk. If you were to type SAVE right now, the current version of the letter on your screen will replace the LETTER file on disk. Instead, let's call the letter on the screen something new, so that the original LETTER remains intact as well. Try typing this:

1/1 WHAT NEXT? RENAME MYLETTER SAVE (*return*)

(You can also do this with the other files we'll be using. That way, you can use the tutorial files as starting places for your own work until you get more familiar with T/Maker.)

How can you check to see that T/Maker actually saved your new MYLETTER chalkboard on the disk? Let's ask T/Maker what files it has on drive A. Type:

1/1 WHAT NEXT? FILES A (*return*)

(Any key you hit now will return you to the WHAT NEXT? prompt.)

The FILES command displays a directory of files contained on the disk. Can you see the chalkboard MYLETTER listed among those files on the disk?

Here's a useful safety feature of T/Maker: Let's save MYLETTER again, and see what happens. Type the following command:

1/1 WHAT NEXT? SAVE FILES A (*return*)

Notice that now there is also a file called MYLETTER.BAK Before T/Maker overwrites the MYLETTER file to save the latest version, it

takes the last version saved and renames it MYLETTER.BAK. So in most cases, if you accidentally save a jumbled version of your file, you can get the back-up copy of the prior version.

Review

So far, we've learned a bit about how T/Maker works.

We've discussed operating T/Maker -- getting files and moving around inside of them.

We compared T/Maker to an *electronic chalkboard*. We can stand *away* from the chalkboard and give T/Maker *commands* at the WHAT NEXT? prompt. Commands generally *do* something to the current working file.

The next half of this tutorial will concentrate on how you can use T/Maker as a solution to typical problems.

Storing Names and Addresses

Let's take a look at the way T/Maker can help us store the names and addresses of our business clients.

First, let's introduce the concept of a database.

Believe it or not, you work with databases everyday. A phonebook is an example of a database. An address book is another example of a database. So is your monthly checking account statement from the bank, or a box of recipe cards.

These real-world databases simply store and present information in a very structured way. For example, you can easily run your finger down the *name* column of a phone book, or the *check-number* column of a bank statement. Databases simply store this type of information in a

pre-described or organized way. Storing information in such a way makes it far easier to find, sort and summarize the information. And, as you will see, even more sophisticated manipulations can be done once information is organized into a database.

Let's look at an address book of clients. Get the chalkboard containing this address-database. Type GET ADDRESS at the WHAT NEXT? prompt.

1/1 WHAT NEXT? **GET ADDRESS** (*return*)

You should be looking at a T/Maker *Record Definition*. The record definition tells T/Maker what information we want to keep track of and how that information should be stored.

It starts with <RECORD>, ends with <END>, and has the information we want stored contained in data *fields* (the things sandwiched in braces {}).

We used EDIT to create the record definition you see in this file. But our good old editor is not as useful for entering information -- there is a better way. Instead, we can use a type of editor designed specifically for working with this database, the UPDATE editor.

When we use UPDATE, we will only be allowed access to the places where information may be stored, that is, inside the fields. Let's try out the UPDATE command. Type this:

1/1 WHAT NEXT? **UPDATE** (*return*)

Again the prompt disappears, but instead of simply entering the file, you are now presented with the first set of information in a more structured format. This address-database has only three clients in it.

Let's take a look at them.

(At this point, you may want to glance at the UPDATE keystrokes listed in the Quick Reference pages.)

Let's try using the UPDATE keystrokes to run around inside this address-database. Press the following keystrokes:

PC VERSION	NON-PC VERSION	ACTION
PG DN	ESC ↓	Jump to the NEXT Record
PG UP	ESC ↑	Jump back to the PREVIOUS Record
ESC ↓	ESC Z	Jump to LAST Record
ESC HOME	ESC A	Jump to FIRST Record
F5	ESC ' ' (Search String)	Search for String, like: 1/1 FIND WHAT STRING? Barker (return)
ESC F9	ESC >	Insert a NEW Record

You'll notice that moving around a database is very similar to moving around in the editor.

Now enter the following name to the database:

**Edward
Slate
Student
353 Campus Dr.
Stanford
CA
94305**

You exit UPDATE just like you would exit the editor -- by pressing the *Exit the Editor* keystroke. Press:

F1 (*Exit UPDATE Editor*)

CTRL-Q

See the T/Maker WHAT NEXT? prompt appear at the bottom of the screen? you've stepped back from the current chalkboard again.

Now you can easily sort or rearrange the order of the address-database by using the T/Maker ORDER command. Let's put our file in alphabetical order. We'll ORDER on the field known as LASTNAME. Type the following:

1/1 WHAT NEXT? ORDER LASTNAME UPDATE (return)

(*Note: Capitalization is very important with database field names. Just as you learned with REPLACE and SEARCH, T/Maker will consider Lastname and LASTNAME to be two different field names.*)

Now let's move around in the address-database to see that T/Maker really has placed the names in alphabetical order.

If we wanted to SAVE this address-database in this new order, we would have to tell T/Maker to SAVE before going on to a new working chalkboard. Jump out of the UPDATE mode now by pressing: -

F1

(Exit UPDATE Editor)

CTRL-Q

Creating a Form Letter

Suppose we'd like to send the LETTER that we worked with in the first half of this tutorial to all our clients in the ADDRESS database. Let's look at two ways we could do this.

Form Letter Method A:

We could get our original LETTER and convert it into a Record Definition. An example of this is filed under FORM.A . Let's get this file. Type:

1/1 WHAT NEXT? **GET FORM.A** (*return*)

We'll want to select information from our ADDRESS file and place it in our form letter on the screen. We'll use the SELECT command to do this. Now type:

1/1 WHAT NEXT? **SELECT ADDRESS END** (*return*)

Let's jump into the editor and see our letters. Type EDIT at the prompt:

1/1 WHAT NEXT EDIT (*return*)

Move down a page at a time to view the form letters which have been created via the SELECT process. Press:

PG DN

(*Screen Down*)

ESC 

Form Letter Method B:

Method A works well for small mail-outs. For larger mail-outs, we'll want to use a more efficient method. If you haven't done so already, Exit the UPDATE mode (you remember -- same keystroke as Exiting the Editor) and get the file called FORM.B:

1/1 WHAT NEXT? GET FORM.B (*return*)

It looks like a mailing label -- with one exception. Notice the *print design command* ".continue LETTER" above the <End> marker.

When we decide to PRINT this file, the print design command tells T/Maker to get whatever is in the file LETTER and also print it out with each record (that is, name and address) in the database.

Let's try creating some form letters. First, we have to SELECT the appropriate information from our ADDRESS file. Type:

1/1 WHAT NEXT? SELECT ADDRESS END (*return*)

Next, let's tell T/Maker to print the current working file. Don't worry if you're not attached to a printer. We're going to have T/Maker print this file to the screen. Type:

1/1 WHAT NEXT? **PRINT IT** (*return*)

"IT" tells T/Maker to print the current chalkboard, rather than a file on disk.

T/Maker will ask you where you would like it printed. For this tutorial, we're going to print to the *screen*. To print to the screen, type **S** at the printer prompt. (*IMPORTANT: You MUST answer "S" to each printer prompt. There will be a total of 5 pages to print.*)

Here is what you will see:

NEXT PAGE 1 (YES.SCREEN.NO.GO.QUIT)? **S**

The first two screens were just a printing of our Record Definition. We could have easily avoided printing these screens by telling T/Maker to begin printing from page 3.

Remember, as we did not **SAVE** that letter in its **ALIGNED** form, it is not going to look as neat as it should. In your own applications, you would normally have **SAVED** the letter in its **ALIGNED** form first.

If you want to pause the printing in progress, just press any key. T/Maker will stop printing to the screen until you press another key to continue.

Mailing Labels

You can easily modify FORM.B to create mailing labels. To see an example of this, get the file called LABEL. Type:

1/1 WHAT NEXT? GET LABEL (return)

Of course, the saluation and the .continue LETTER design command are both now gone for the purposes of a mailing label. A new print design command is included at the top of the Record Definition -- .pagesize 14. It tells T/Maker that we are using mailing labels with 14 lines per label.

The same command we used before will make a set of mailing labels. Try this:

1/1 WHAT NEXT? SELECT ADDRESS END (return)

You can jump into the editor to look at your labels, or go through the PRINT function and print them to the screen to look at them.

Creating An Invoice

Going through a simple invoicing example may be a good way to show off the math capabilities of T/Maker's database.

Let's get the example invoice contained on disk. Type:

1/1 WHAT NEXT? GET BILLING (*return*)

Notice the *Rules Definition* located above the Record Definition. The rules definition allows us to alter the value of a given data field.

Notice that the area for TOTAL is not yet filled in. Let's set these rules in motion and see what happens to our invoice-database. We'll use the T/Maker SET command. Type:

1/1 WHAT NEXT? SET (*return*)

Take a look at what happened. T/Maker figured out the TOTAL value for us.

We can easily set one of the values equal to a constant by adding an extra relationship in our Rules Definition. Let's try setting PRICE = 100.

First, let's jump into the Editor. Type EDIT at the WHAT NEXT? prompt.

1/1 WHAT NEXT? EDIT (*return*)

Move the cursor down until it rests on the first line in the Rules Definition (on the "TOTAL" line). Insert a line here by PRESSING:

F9

(*Insert a Line*)

CTRL-R

Type in the following relationship (make sure "PRICE" is CAPITALIZED - remember, capitalization is important when naming database fields.)

<Rules>

PRICE = 100 *-- you insert this line*

TOTAL = PRICE * QTY

<END>

Now Home the cursor by pressing:

HOME

(Home the Cursor)

CTRL-O

And exit the editor by pressing:

F1

(Exit the Editor)

CTRL-Q

And finally, set your new Rules in motion by giving T/Maker the SET command:

1/1 WHAT NEXT? **SET** *(return)*

Doing math from within T/Maker's database puts a lot of business power at your finger tips. But there's another way to do it:

Spreadsheet Analysis

A *spreadsheet* is nothing more than a calculator that operates on a row, column, or table of numbers.

T/Maker does spreadsheet analysis in a very *visible* way. T/Maker *calculates by example* -- that is, by supplying T/Maker with an example of where columns of number are located, and, of course, with the desired equations, T/Maker can act as a visible table calculator!

Let's take a look at this visible approach to spreadsheet analysis.
Type:

1/1 WHAT NEXT? GET TABLE (return)

The top line of this spreadsheet begins with the word EXAMPLE. This line is very important to our spreadsheet: any sets of characters placed on the *Example Line* represent a place where we will have columns of numbers.

Below the Example Line is a line which begins with AC1. This *Row Equation* which tells T/Maker to add the numbers in the first and second column and put the result in the third column.

The area *under* the word EXAMPLE is a seven-column section reserved for equations. Notice that the *Column Equation* in this area looks very much like the output from a 10-key adding machine.

To compute this spreadsheet table, why not ask T/Maker to COMPUTE?

1/1 WHAT NEXT? COMPUTE (return)

One of the benefits of organizing numbers in a spreadsheet is that we can then play "what if" with our numbers. For instance, what if Rubble had only purchased the same dollar amount in February as he had in January. What would that mean to our bottom line?

Let's check it out. First, let's jump into the Editor.

1/1 WHAT NEXT? EDIT (*return*)

Move the cursor down to the first row of numbers and change the second entry to 250 by typing over what is already there (be sure to stay in the "B,BBB" column!) Your Rubble entry line should now look like this:

+ Rubble 250 250

Now, press the keystrokes to home the screen and exit the Editor:

HOME

(*Home the Cursor*)

CTRL-O

F1

(*Exit the Editor*)

CTRL-Q

Finally, type COMPUTE at the WHAT NEXT? prompt:

1/1 WHAT NEXT? COMPUTE (*return*)

T/Maker computes our new total. Good thing Rubble increased his purchases!

We can also add another customer in a flash. Enter the Editor (by typing **EDIT**) and space down so that your cursor rests on the line for Barker. First we'll make some room for a new entry. Press:

F9

(Insert a Line)

CTRL-R

Next, put a plus sign in the first column of this new line, so it lines up with the other plus signs already there. Space over to the name column, and type in a name of your choice on this line. Now space under the other numbers and type in 500 for January and 1000 for February. The line you have just added should look like this:

+ My Name 500 1000

Now let's compute our new table, with the same steps we used before. Press:

HOME

(Home the Cursor)

CTRL-O

F1

(Exit the Editor)

CTRL-Q

Finally, type **COMPUTE** at the **WHAT NEXT?** prompt:

1/1 **WHAT NEXT? COMPUTE** (*return*)

T/Maker computes our new total. Notice it also added the comma in the 1,000 entry.

We may want to print this table in a format which does not include all the math notation. We can use the CLEAN command clean up our table for printing. Type:

1/1 WHAT NEXT? **CLEAN** (*return*)

Publication quality! (*NOTE: You normally want to save your spreadsheets with the equations intact, so that you can return to them and change them whenever you want to. You'd usually SAVE your spreadsheet before you CLEAN it, and never save the cleaned version. Another way to CLEAN the spreadsheet is with a .CLEAN Print Design Command. That way, it always prints out cleaned, but is always stored in original format.*

Peeking at the Rest of T/Maker

We've experimented a bit with T/Maker's Word Processor, DataBase Manager, and SpreadSheet. In each case, we tried to keep it simple, taking you through some basic functions. However, once you've successfully completed this tutorial, you should find the rest of T/Maker to be a logical extension of what you already know.

In addition to what you have already sampled, the word processor, database and spreadsheet functions have many sophisticated features. A look through the manual and the HELP files should convince you of this.

There are other features and commands we never even touched upon! For example, with T/Maker's Spelling Checker, you could spell check your form letter for typos or misspellings and have T/Maker present you with spelling options just by asking. You could use the MIX command to encrypt a "top-secret" file. And, the Systems Design features can help you build sophisticated, automated systems.

We hope you enjoyed the tutorial, and are excited about the possibilities before you . . . we hope T/Maker will become the key program in your software library -- or even the only software program!

FILE MANAGEMENT COMMANDS

This chapter describes the nineteen T/Maker commands designed to manage files:

```
=====
T/MAKER'S FILE MANAGEMENT COMMANDS
=====
CREATE    creates a working file with a unique name.
SAVE      stores the working file on disk.
GET       retrieves a file from disk, making it the working
          file.
DELETE    discards a file on disk.
RENAME    lets you give the working file a new name.
INSERT    lets you add the contents of a disk file to the
          working file.
MERGE    lets you overlay the contents of a disk file
          on the working file.
LIST     lets you review the contents of a disk file
          without displacing the current working file.
CLIP      discards portions of the working file.
FILES    screens a directory of files on disk.
DATA     sets the default disk drive.
PATH     sets the directory for data files (MSDOS 2.0 only).
RESET    resets disk drives.
1/1      lets you place the screen at the desired line and
          column position in the file.
INFO     temporarily displays a variety of facts about the
          working file.
TABS     lets you save tab settings with a file.
NOTABS   lets you discard tab settings when file is saved.
WAIT     lets you interrupt a series of T/Maker commands.
STOP     terminates T/Maker.
=====
```

The chapter has three sections. The first section provides a brief discussion of T/Maker's conventions or rules for naming files. You need to know the limits of legal filenames when you are using T/Maker's CREATE and RENAME commands.

The chapter's second section describes four fundamental file management commands: CREATE, SAVE, GET, and DELETE.

The chapter's third section describes the remaining fourteen file management commands.

TABLE OF CONTENTS

SECTION ONE: Rules for Naming Files

File Names	FM-1
------------------	------

SECTION TWO: The Fundamental Four

CREATE	FM-2
SAVE	FM-2
GET	FM-3
DELETE	FM-4

SECTION THREE: The Remaining Fourteen

RENAME	FM-5
INSERT	FM-5
MERGE	FM-5
LIST	FM-6
CLIP	FM-6
FILES	FM-6
DATA	FM-7
PATH	FM-8
RESET	FM-9
{Line-number/Position-number}	FM-10
INFO	FM-10
TABS/NOTABS	FM-11
WAIT	FM-12
STOP [& KILL CHARACTER]	FM-12

SECTION ONE: RULES FOR NAMING FILES

You assign a name to a file when you CREATE or RENAME one. If such a file is later SAVED, then it will be known by the name you have assigned to it. You will also use this name when you retrieve the file from disk with T/Maker's GET command.

There are a few conventions and rules you must observe in naming files. All filenames have two parts:

MAIN NAME
FILE TYPE

The "main name" is comprised of a series of up to eight characters. The "file type" is comprised of a period followed by up to three characters.

Generally speaking, the characters used in a filename should be letters or numbers. Special characters--for example, punctuation marks or other keyboard symbols--should be avoided. Your operating system documentation will give you precise rules for composing file names.

It is not necessary to specify a "file type."

Here are some examples of valid filenames:

GROSS	
SALES82.DPA	(SALES82/DPA for TRSDOS)
PERSONAL.LET	(PERSONAL/LET for TRSDOS)

You may type filenames in either upper-, lower-, or mixed-case letters.

At times you may want to specify a disk-drive in a filename. When you do not specify a disk-drive, T/Maker assumes you are naming a file on the "default drive." T/Maker's DATA command (see below) lets you establish the default disk drive.

You override the default-drive assumption by incorporating a disk-drive designation in the name. like this:

a:budget.jun (budget/jun:0 for TRSDOS)

SECTION TWO: THE FUNDAMENTAL FOUR

This section describes four fundamental file management commands: CREATE, SAVE, GET, and DELETE. These commands are essential to virtually all T/Maker operations.

CREATE

The CREATE command establishes a new working file and assigns a name to it. You use this command by typing "CREATE" followed by the filename you wish to assign the new file. Then you press the RETURN key. The command looks like this:

WHAT NEXT? CREATE file-name

Important Note: All T/Maker commands that are typed-in as responses to the WHAT NEXT?-prompt require that you press the RETURN key to put them into effect. For brevity's sake, we will not cite the necessity for pressing the RETURN key any further in the descriptions of commands and their use.

When, at a later point, you save this file on disk, the name you have assigned to the file becomes the name you use to retrieve it from disk, using T/Maker's GET command.

Nothing is stored on disk simply by virtue of having used the CREATE command.

T/Maker prohibits you from assigning a name that is already in use as the name of file currently saved on the relevant disk-drive.

SAVE

The SAVE command stores the working file on disk under the name you have assigned it. You use this command by typing SAVE or S. The SAVE command always stores the entire working file on disk.

WHAT NEXT? SAVE or S

Unnamed Files. If you did not happen to assign a name to your working file (say, because you went directly into the Editor

rather than first using the CREATE command), T/Maker automatically assigns the file the name TMAKER.TXT to it. You can change this name at any point by using the RENAME command (see below).

Automatic Backup of Files. T/Maker automatically backs-up stored files. An example will best serve to clarify T/Maker's backup procedure:

Suppose that you create a new file named LETTER.1 and use the SAVE command to store it on disk. Now, the working file is called LETTER.1 and there is a file on disk with the same name.

Further suppose that you revise the working file in some way and then use the SAVE command on this file again. This second use of the SAVE command will cause T/Maker automatically to change the name of the already-stored version of LETTER.1 from LETTER.1 to LETTER.BAK. The current working file will be stored under the name LETTER.1.

Finally, suppose that you revise the working file and the SAVE command again. This time T/Maker would (1) delete the disk file named LETTER.BAK, (2) change the name of LETTER.1 to LETTER.BAK, and (3) store the current working file under the filename, LETTER.1.

In this way, T/Maker always maintains one level of storage backup. Note, however, that two different files, one named named LETTER.1 and the other named LETTER.2, will have only one backup file, LETTER.BAK, between them.

Storing Tab Settings. The SAVE command will not necessarily store a file's tab settings with the file. You use the TABS or NOTABS to specify explicitly whether tab settings should be so stored. T/Modify's Option 3, described in the Customization Notes, can be used to set T/Maker's default choice concerning automatic tab storage. In general, it is convenient to store tab settings with files. If, however, you are likely to do work on a file with programs other than T/Maker, it is best not to store tab settings. T/Maker's method of encoding tabs into files often will be incompatible with other programs.

GET

The GET command loads a file from disk into memory, establishing the file named as the working file.

WHAT NEXT? GET or G filename

If there is a working file already in the computer's memory and you use the GET command, the newly gotten file displaces the pre-existing working file. Therefore, you should take care to

save current working files you wish to preserve before using the GET command to establish a new working file. You can use the LIST command (see below) to examine a disk file without displacing the current working file.

DELETE

The DELETE command deletes a file from disk.

WHAT NEXT? DELETE disk filename

If it happens that the filename you ask T/Maker to delete does not actually exist on disk, no error message is generated.

SECTION THREE: THE REMAINING FOURTEEN

RENAME

The RENAME command gives a new name to the working file. A SAVE command will thereafter save the working file under this new name.

WHAT NEXT? RENAME new filename

The RENAME command does not affect the status of the file with the old name remaining on disk. Typically, the RENAME command is used in situations where it is desirable to make changes to an existing file or create a different version of it, while still preserving the original. One simply gets the existing file, renames it, and then proceeds to make modifications.

INSERT

The INSERT command inserts a file on disk at the top of the active area. Pre-existing text in the file is pushed down to make room for the inserted text.

WHAT NEXT? INSERT filename

Tip: It may be convenient for you to keep a supply of small files containing frequently used conventions on disk. For example, a file might be built containing the aligning wedges and top or bottom routines for a typical business letter. Then, when starting a new letter, the file of conventions can be inserted into the new file.

MERGE

The MERGE command overlays a disk file upon the working area.

WHAT NEXT? MERGE disk-file-name position

A combination of the screen and a specified parameter tell T/Maker where upon the working file to overlay the disk file. The top of the screen provides the line for the top line of the disk file; the column-position for column-position 1 of the disk file is specified in the command as a digit following the command-word "MERGE."

For example, suppose a file on disk named "BOX" were a neatly drawn box. Say that the upper left hand corner of this box lay at line 1, column-position 1 in the file. Suppose that you desired to superimpose this box over some text in the file so that the upper-left hand corner of the box lay at line 10, column-position 15 in the working file. How would you instruct T/Maker's MERGE command?

First, you would make line 10 of the working file the top line on the screen. This you could accomplish either with the Line Number/Column Number command or by entering the editor and moving the cursor to the tenth line and then quitting the editor. Then, you would give the following command:

WHAT NEXT? MERGE BOX 15

LIST

The list command lets you screen a disk file without displacing the current working file.

WHAT NEXT? LIST disk-file-name

The listed file scrolls on your screen. This scrolling can be temporarily interrupted and then restarted simply by touching the spacebar. Once the disk file is completely displayed, touching the spacebar will return the WHAT NEXT?-prompt to your screen. (See also the Print command for printing a disk file on the screen.)

CLIP

The CLIP command discards that portion of the working file lying before or after the top line on the screen (when you are outside the Editor).

WHAT NEXT? CLIP { BEFORE or AFTER }

"Before" and "after" may be abbreviated to "b" or "a".

FILES

The FILES command temporarily screens a directory of disk files. The working file is overlaid but not displaced by this directory. You specify the disk drive whose directory you want to see by providing the drive's designation after typing FILES, like

this:

WHAT NEXT? FILES disk-drive-name

Long directories of files will be scrolled on your screen. You can temporarily interrupt this scrolling by touching any key; thereafter, touching a key will restart the scrolling.

Once the directory of files has been completely presented, touching any key brings the WHAT NEXT?-prompt to the screen.

The FILES command does not alter the working file even though it preempts the screen's display temporarily. You can return the working file to the screen by typing E or the number that appears at the bottom of the screen next to the WHAT NEXT?-prompt.

T/Maker follows a simple convention in the presentation of filenames in the directory. Files are grouped by file-type (does not apply to TRSDOS)--that is, by the second part of the filename, the part following the period. Knowing this, you may wish to invent filenames that generate a well ordered directory.

For example, suppose you had one disk devoted to your correspondence with ten customers. Every piece of correspondence is lodged in its own file. In this case, you might name files so that correspondence to the same customer received the same file-type name: LETTER1.BMN, LETTER2.BMN, INVOICE1.BMN, and COMPLNT1.BMN. In this way, all files relating to customer BMR would be listed together in the T/Maker's directory.

For MSDOS and CP/M 86, wildcard characters can be used in a FILES command. For example, "FILES B:*.BAK" will list all files no drive B with a file-type of ".BAK".

DATA

The DATA command lets you establish T/Maker's default disk drive.

WHAT NEXT? DATA drive-name

Once you give this command, filenames lacking disk drive designations will be assumed to refer to the default drive. This is a handy way to avoid having to type "B:" or ":1" (TRSDOS) with filenames.

The default drive can be overridden at any time by specifying a disk drive in your command. You must merely add the disk drive prefix to your command--for example, "GET A:TEST.1".

To return to the condition of no default drive except the system defaults, give a space surrounded by quotation marks as the drive name.

DATA '' (DO NOT USE FOR TRSDOS)

To establish a fixed default drive--so that T/Maker automatically uses a specified default drive each time it is used, see Customization Notes, T/Modify Option 2.

PATH (MSDOS 2.0 only)

The PATH command is used to move among directories. You should consult your MSDOS manuals for details on directories and make use of MSDOS commands to create or delete them. T/Maker's PATH command behaves in a similar way to the MSDOS "cd" command.

For floppy diskettes systems, we recommend that you do not use directories as this will only slow the process of using T/Maker. You may also create some confusion and errors if you switch any diskettes while running T/Maker.

For a hard disk, we suggest that you keep your directory structure as simple as possible since you cannot create, list, or otherwise deal with directories from within T/Maker. Relying on simple names you can remember and perhaps only one level of directories would appear an appropriate strategy.

On the hard disk, you must keep all of T/MAKER (including commands, ".UTL" files, and dictionaries in one directory). After starting T/Maker from within this directory, you may effectively get into another directory by using the PATH command.

WHAT NEXT? PATH E:\TABLES

The above command would set the directory for the "E" drive to "\tables." This does not, however, change the Default Drive for Data which is set with the DATA Command. You must make a separate use of the DATA command if you want to set the default drive.

GET, SAVE, FILES, and other commands made in reference to drive "E" would be assumed to apply to this directory.

WHAT NEXT? PATH \JOE\REPORT

As shown above, the drive designator need not be specified if you want to move to another directory on the Default Data Drive.

Whenever the PATH command is used, the new path is displayed at the top of the screen. You can find out the current path for a given drive by using the PATH command with only a drive designator.

WHAT NEXT? PATH A:

You may also use the conventions of the MSODS CD command to move up one directory.

WHAT NEXT? PATH ..

While the Drive is a formal part of the name of a file in T/MAKER, the directory can be changed independent of the file name and is not a part of it. Thus, you should not change the path on your data drive after you GET a file but before you SAVE it unless your intent is actually to copy the file from one directory to the other on the same drive.

In general, changing directories should represent a complete switch from one activity to another with almost no need to deal with files in two different directories for a single task.

RESET (CP/M or CP/M 86 only)

You must use the RESET command before you attempt to write on or delete a file from a newly inserted disk. You simply type the command word after changing disks.

WHAT NEXT? RESET

If the DELETE or SAVE commands are used without first using the RESET command an error will occur and the working file will be lost. This error produces the message "BDOS R/O" from your operating system.

If you happen to change disks frequently, you may want to tell T/Maker to impose a RESET command automatically before it executes any of your SAVE or DELETE commands. This is done by using T/Modify's Option 3, described in the Customization Notes.

(LINE NUMBER) (/COLUMN POSITION-NUMBER)

This command moves the upper left hand corner of the screen (or the angle-point of the frame, when you are in frame mode) to the designated coordinates, without you having to enter the editor. You simply type in the desired coordinates after the WHAT NEXT?-prompt. You can specify the line number alone, the column position alone, or both the line number and column positions. The number designating the column position is always preceded by a slash (/).

WHAT NEXT? 25 or 35/40 or /150

This command can also be used to redraw the working file after the display has been preempted by some other command. To accomplish this, simply type the line number that already appears next to the WHAT NEXT?-prompt in response to the prompt.

INFO

The INFO command displays information about the working file and places a column ruler on the screen.

WHAT NEXT? INFO

This command, which is used when you are outside the editor, produces the same result as the "Information" keystrokes, which is used when you are inside the editor. A display like the following is shown on your screen:

```
NAME: B:JONES.TAB LINES: 93 FREE: 11804 LINE: 18/1 MODE: M/I
.....1.....2.....3.....4.....5.....6.....
12345678901234567890123456789012345678901234567890123456
```

These are called the INFO lines.

Like the FILES command, the INFO command temporarily overlays, but does not displace, the working file.

The lower two INFO lines provide a ruler for the screen. The ruler should be read "vertically"--which is to say, the numbers in the third line report the\unit\numbers and the numbers in the next line up report the\tens\numbers. This ruler can be handy for setting tabs and other editorial and data-processing needs.

The top INFO line reports a number of pieces of information:

NAME: B:JONES.TAB LINES: 93 FREE: 11804 LINE: 18/1 MODE: M/I

READING THE TOP INFO LINE:

Line Segment	Content
1. "NAME: B:JONES.TAB"	File name and location: The working file is "JONES.TAB" on the B disk drive.
2. "LINES: 93"	File length: The file is 93 lines long.
3. "FREE: 11804"	Free characters remaining in the file: 11,804 remaining.
4. "LINE: 18/1"	The cursor's location when INFO was asked: line 18, position 1.
5. "MODE: M/I"	Editing Modes (see INFORMATION Key-stroke in the Editor).

TABS/NOTABS

The TABS command causes subsequent SAVE commands to save a working file's tab settings with the file being saved (see also description of the SAVE command for further information).

WHAT NEXT? TABS

The Notabs command tells T/Maker's SAVE command not to save tab settings with the working file.

WHAT NEXT? NOTABS

WAIT

The WAIT command suspends the processing of a series of commands until a key is touched at the console. This command might be used in a series of prespecified commands where it is necessary to change disks, like this:

WHAT NEXT? SORT 1 10 WAIT RESET SELECT ADDRESSES END

STOP [& KILL CHARACTER]

The STOP command discontinues the use of T/Maker and returns your computer to the operating system.

WHAT NEXT? STOP

T/Maker also provides a "kill keystroke." This keystroke--which varies depending upon the kind of computer you have--can halt whatever is T/Maker is currently doing.

On most T/Makers, the kill keystrokes are either <CTRL> Q or <ESC> .

The kill keystroke can be changed with T/Modify Option 2, described in the Customization Notes.

WORD PROCESSING

Word processing with T/Maker involves three major commands. These are Edit (used to create or revise text), Align (used to justify or center text), and Print (used to print text and achieve special presentation requirements).

Naturally, Edit and Print are used in other application areas as well.

TABLE OF CONTENTS

THE EDIT COMMAND	WP-1
Entering the Editor	WP-2
Inside the Editor	WP-2
How to use the <ESC> and <CTRL> keys	WP-3
Quitting the Editor	WP-3
Moving the Cursor	WP-4
Carriage Return Modes	WP-8
Revising Text	WP-9
Insert Mode	WP-13
Finding and Replacing Strings	WP-14
Tabs	WP-16
Moving Lines	WP-17
Keystroke Macro	WP-18
Redrawing the Screen	WP-21
The Info Command	WP-22
Moving Columns	WP-24
Frame Mode	WP-26
THE ALIGN COMMAND	WP-28
Basic Aligning Options	WP-29
Block Text	WP-30
Paragraph Demarcation	WP-31
Indenting or Numbering Paragraphs	WP-32
Centering	WP-33
Stop Aligning	WP-33
Aligning Shortcuts	WP-35
Single-side Changes	WP-35
Tagged and Recalled Wedges	WP-35
Exact Spacing Within Aligned Text	WP-37

THE PRINT COMMAND	WP-38
To Print the Working File	WP-39
The Printer's Prompt	WP-40
Printing to the Screen	WP-41
The Print Command, More Generally	WP-42
Stringing and Stacking Print Commands	WP-44
Design Commands	WP-45
Single or Double-Spaced Text	WP-46
Blocking Text	WP-47
Forced New Page	WP-49
Optional New Page	WP-49
Widowing Lines	WP-49
Start Text on Line "X"	WP-50
Top and Bottom Routines	WP-51
Page-Headings and Page-Endings	WP-51
Placing Top and Bottom Routines in a Document	WP-53
Changing Top and Bottom Routines	WP-54
Delaying the Use of Top and Bottom Routines	WP-55
Other Facts about Top and Bottom Routines	WP-55
Numbering Pages	WP-56
Arbitrary Page Numbers	WP-57
Footnotes	WP-58
Bold and Underscored Text	WP-59
Organizing and Printing Long Documents	WP-61
The Continue Command's Governing Principle	WP-63
The Physical Page	WP-64
Page Size	WP-64
Page Length	WP-65
Page Width	WP-66
Other Design Commands	WP-66
Indenting Blocks of Text	WP-66
Comment Lines	WP-67
Non-Lines	WP-67
Non-Aligning Line	WP-67
"Cleaning" Tables	WP-68
Printing Aligning Wedges	WP-68
WORD PROCESSING EXAMPLES	WP-69
A Multi-Page Letter	WP-69
A Simple Approach to Form Letters	WP-70
A Long Document (Master File)	WP-71
A Long Document (Text File)	WP-72
Mixing Text and Tables	WP-73

THE EDIT COMMAND

The Edit command is the heart of T/Maker's word processing system.

Fundamentally, when working inside the editor you are placing on the screen an image of the document you want to produce.

The edit command is T/Maker's way of providing you with a blank slate upon which to write and revise. It makes your screen into that slate.

Note: You will need to use a number of other commands in conjunction with the Edit command. Mostly, these are file management commands.

For example, you will want to CREATE a document and a name for it, so that your document can be stored on disk and recalled when wanted. Once typed up in the editor, you will want to SAVE the document, with the SAVE command. Now and then you may want to RENAME a document.

For more detail on these sorts of commands, see the "File Management and Other Commands" Section of this manual.

ENTERING THE EDITOR

You enter the editor by typing "edit" or "e", for short, (and a carriage return) in answer to the Universal Prompt.

1/1 WHAT NEXT? E or EDIT

You are now ready to "edit a file," which means you are ready to type a new document or revise an old one. If creating a new document, you will have used the CREATE command before entering the editor, to give the new document a name. If you are working on an old document, you will have used the GET command to retrieve that document from disk.

When you are in the editor, there is no "WHAT NEXT?" prompt on the screen. To get the prompt back on the screen, you have to "quit the editor" (described later).

INSIDE THE EDITOR

When you are outside the editor, you run T/Maker by giving one or another answer to the "WHAT NEXT?" prompt. But when you are working inside the editor, the prompt is not on the screen any more. So, when inside the editor, you will need a different sort of method for telling T/Maker what to do.

Inside the editor, touching a key on the keyboard has the effect of putting that key's image on the screen. That is as might be expected.

You can also use keyboard keys in a way that does not produce screen images but instead gives commands or instructions. This is done by using two special keys in conjunction with the regular keyboard keys, the CONTROL KEY (<CTRL>) and the ESCAPE KEY (<ESC>). For example, an alphabet key touched in conjunction with a <CTRL> or <ESC> key does not appear on the screen, and will often have a special command function, say, moving the cursor up or down.

Much of what is described in this part of the manual concerns these controls you exercise while in the editor.

You should note that in some cases two different keystrokes are given. One set is marked "PC" and should be used for IBM PC compatible machines using MSDOS. The other set marked "NON-PC" should be used in other cases. Always consult your Addendum to see if special keystrokes (i.e. neither PC nor NON-PC) have been used for your computer.

HOW TO USE THE <ESC> AND <CTRL> KEYS:

The <ESC> and the <CTRL> keys are always used in conjunction with some other key. They are used--in the sense of "physically manipulated"--a bit differently from each other.

When the <CTRL> key is used, it is HELD DOWN while you touch another key. So, for example,

<CTRL> P

means "hold down the Control Key and touch the P key" (either a capital or small "p" will do, by the way).

When the <ESC> key is used, it is simply touched, and then the additional key is touched. So,

<ESC> P

means "touch the Escape Key (release it) and touch the P key".

Not infrequently commands call for both the <ESC> key and the <CTRL> key, as in,

<ESC> <CTRL> P

This means, "touch the Escape Key and then hold down the Control Key and touch the P key".

QUITTING THE EDITOR

You leave or quit the editor by typing <F1> (Function Key 1) for the PC or "<CTRL> Q" otherwise--this brings back to the screen the "WHAT NEXT?" prompt.

The line the cursor lies on at the time you quit the editor immediately becomes the top line on the screen.

Note: The top line on the screen is an important consideration, because many commands performed while you are outside the editor commence from the screen's top line. An important exception to this rule is the Save command, which saves the whole working file regardless of the screen's top line.

Because of this, you will sometimes want to send the cursor to the origin of your document before quitting the editor (see "Moving the Cursor").

MOVING THE CURSOR

For a moment, think of T/Maker as a typewriter that works on a huge piece of paper--a piece 300 positions wide and very long.

Your screen, of course, isn't big enough to see the whole of the paper at once. So, not only do you need ways to move the cursor on the screen, you need ways to move the screen around on the underlying paper as well.

Incidentally, all these cursor movements leave the text undisturbed.

<u>PC KEYSTROKE</u>	<u>NON-PC KEYSTROKE</u>	<u>MOVES THE CURSOR . . .</u>
-------------------------	-----------------------------	-------------------------------

Up Arrow	<CTRL> K ...	UP
Down Arrow	<CTRL> J ...	DOWN
Left Arrow	<CTRL> H ...	LEFT
Right Arrow	<CTRL> L ...	RIGHT
<PG UP> ...	<ESC> <CTRL> K ...	UP A SCREEN'S HEIGHT
<PG DN> ...	<ESC> <CTRL> J ...	DOWN A SCREEN'S HEIGHT
<ESC> <PG UP> ...	<ESC> <CTRL> H ...	SCREEN MOVES LEFT
<ESC> <PG DN> ...	<ESC> <CTRL> L ...	SCREEN MOVES RIGHT
<HOME>	<CTRL> O ...	TOP LEFT OF SCREEN
<ESC> <HOME> ...	<ESC> <CTRL> O ...	TOP LEFT OF DOCUMENT
<ESC> Up. Arw	<ESC> A ...	FIRST LINE OF DOCUMENT (the position is unchanged)
<ESC> Dn. Arw	<ESC> Z ...	BOTTOM LINE OF DOCUMENT (the position is unchanged)
<END>	<CTRL> P ...	ADVANCE TO THE NEXT WORD
<ESC> <END> ...	<ESC> <CTRL> P ...	ADVANCE TO THE END OF A LINE
<RET>	<RET> ...	CARRIAGE RETURN, TO THE LEFT EDGE OF THE SCREEN
<ESC> <RET>	<ESC> <RET> ...	CARRIAGE RETURN, TO THE DOCUMENT'S LEFT EDGE

Note: Notice that the ESCAPED form of a command is often a sort of extreme version of the UNESCAPED form. For example, <CTRL> K moves the cursor up a line, while <ESC> <CTRL> K moves it up a screen's height.

Two other commands also move the cursor: Tabs and Searching (see the subsections on these commands).

CURSOR MOVEMENTS (PC)

Working File

■ <ESC> <HOME>

■ <ESC> Up Arrow

■ <PG UP>

Screen Left

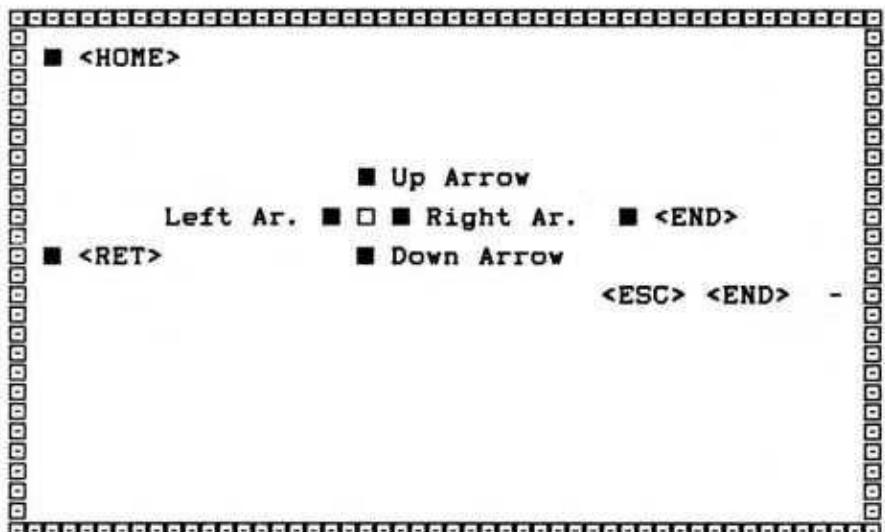
<ESC> <PG DN>

Screen Right

<ESC> <PG UP>

SCREEN

■ <ESC>
<RET>



■ <PG DN>

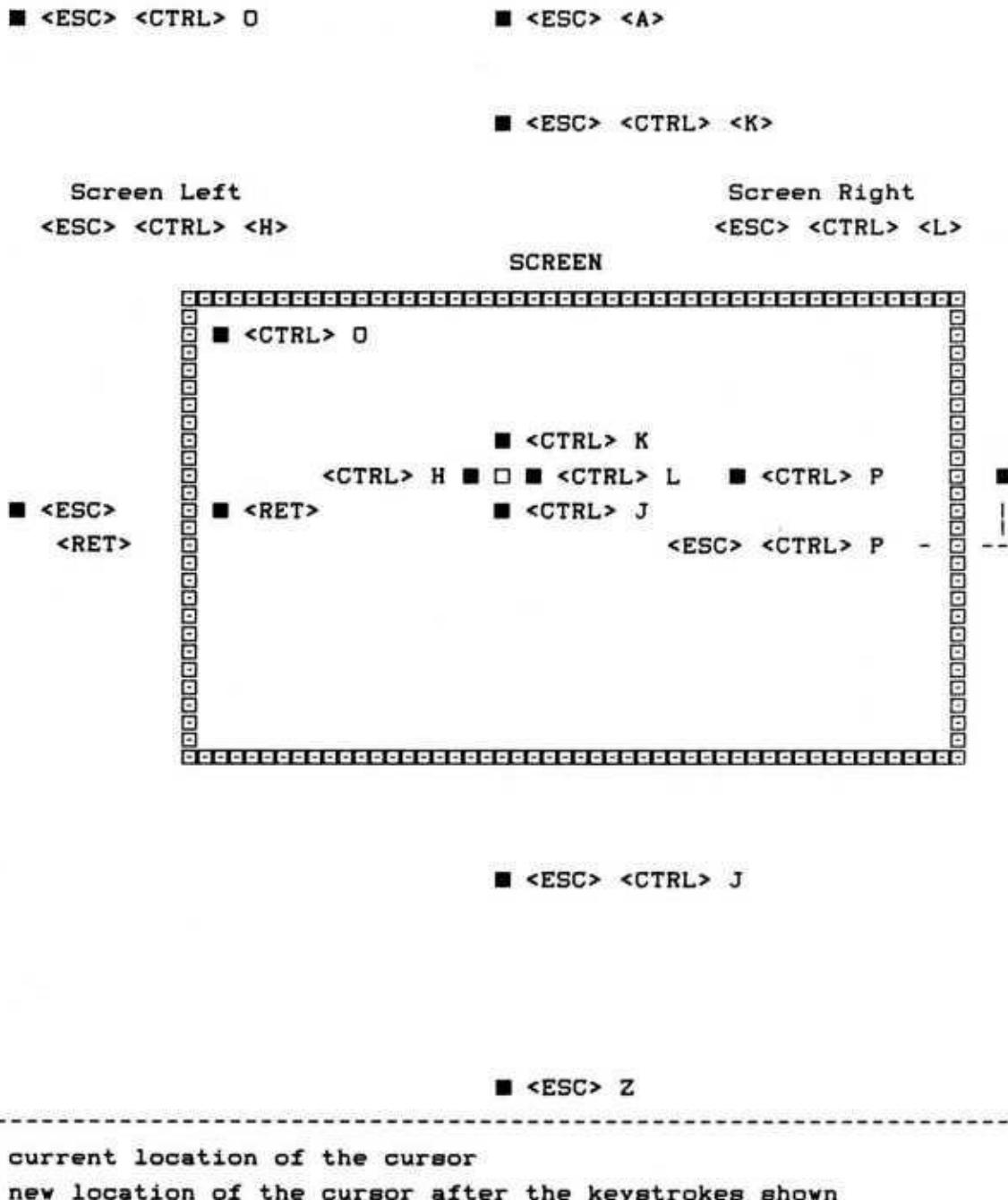
■ <ESC> Down Arrow

□ current location of the cursor

■ new location of the cursor after the keystrokes shown

CURSOR MOVEMENTS (NON-PC)

Working File



There are a few particulars about moving the cursor that are worth keeping in mind:

DOWN .. When moving the cursor down from the bottom line on the screen, the screen will scroll one line forward.

UP .. When moving the cursor up from the top line on the screen, the screen will scroll. Depending on the capabilities of your computer, this will appear as smooth operation or the screen will be entirely redrawn to show the new image.

LEFT .. Moving the cursor to the left from the leftmost position on the screen will do nothing or it will cause the screen to move to the left. Which one happens depends on the setting of Carriage Return Mode (discussed in the next subsection).

RIGHT .. Moving the cursor to the right from the rightmost position on the screen will do nothing or it will cause the screen to move to the right. Which one happens depends on the setting of Carriage Return Mode (discussed in the next subsection).

SCREEN LEFT .. Moving the screen to the left moves a "half-screen" to the left if the cursor is currently in the leftmost or rightmost position of the screen. Otherwise, the screen is moved so that the current position of the cursor becomes the rightmost position on the screen.

SCREEN RIGHT .. Moving the screen to the right moves a "half-screen" to the right if the cursor is currently in the leftmost or rightmost position of the screen. Otherwise, the screen is moved so that the current position of the cursor becomes the leftmost position on the screen.

ADVANCE TO NEXT WORD .. Advancing the cursor to the next word when the next word is off the screen will cause the screen to be moved so that the next word starts in the leftmost position of the screen.

ADVANCE TO END OF A LINE .. Advancing the cursor to the end of the line when the end of the line is off the screen will cause the screen to be moved so that the end of the line sits at the middle of the screen.

CARRIAGE RETURN MODES

T/Maker III gives you three carriage return options. You pick the option you want, and then type. The options are:

1) DRAFTING MODE

This mode is good for drafting text. The carriage automatically returns when the cursor approaches the screen's right side.

2) MANUAL MODE

In this mode the carriage does not automatically return. You return it manually. Otherwise, the cursor comes to a stop at the right edge or left edge of the screen.

3) TABLE MODE

In this mode the carriage does not automatically return. Instead the cursor will continue on past the right edge of the screen on the same line. (The screen repositions itself to show you the cursor again. Left and right cursor movements at the edge of the screen will also cause the screen to move.) This mode is for doing tables or text wider than the screen.

To change the current carriage return mode, touch

<ESC> R

T/Maker works on a revolving principle in this case. Each time you touch **<ESC> R** the carriage return mode changes to the next option, like this:

Manual Mode --> Drafting Mode --> Table Mode --> Manual Mode

Note: In all three modes, you can at any time touch the **<RET>** key to return the carriage to the left edge OF THE SCREEN, a line lower, or, touch **<ESC> <RET>** to return the carriage to the left edge OF THE DOCUMENT, a line lower.

The default carriage return mode is MANUAL MODE.

When the carriage return mode is changed, the Information command (discussed in a subsequent subsection) is automatically invoked, so that you can see the mode you have entered.

REVISING TEXT

The T/Maker editor provides a variety of ways to revise text.

<u>PC KEYSTROKE</u>	<u>NON-PC KEYSTROKE</u>	<u>THE EFFECT . . .</u>
	<CTRL> E	DELETES A CHARACTER
<F10>	<CTRL> T	DELETES A LINE
<ESC> <INS>	<CTRL> U	DELETES ALL BLANK SPACES TO THE NEXT CHARACTER
<ESC> 	<CTRL> Y	ERASES FROM THE CURSOR TO THE END OF A LINE
<BACK>	ERASES TEXT AS THE CURSOR MOVES BACKWARDS
SPACEBAR	SPACEBAR	ERASES TEXT AS THE CURSOR MOVES FORWARD
overtyping ...	overtyping	REPLACES OLD TEXT WITH NEW
<ESC> I	<ESC> I	INSERTS TEXT INTO A LINE
<INS>	<CTRL> W	INSERTS A CHARACTER SPACE
<F9>	<CTRL> R	INSERTS A BLANK LINE
<ESC> <F10>	<ESC> <	JOINS LINES
<ESC> <F9>	<ESC> >	BREAKS LINES

Note: The <ESC> I command works a bit differently from the others--see "Insert Mode" subsection. . ,

HOW TO USE TEXT-REVISING COMMANDS:

DELETION COMMANDS:

**** (<CTRL> E for non-PC) deletes a character. This command is used by moving the cursor "over" the character to be deleted. Then press and the deletion is done.

EXAMPLE: "Rose is a lovely bab3y!" is corrected by pressing when the cursor is over the "3".

<F10> (<CTRL> T for non-PC) deletes a line. This command is used by moving the cursor to any place on the line you want to delete. Then give the command and the deletion is done.

EXAMPLE: With the cursor as shown, the command will delete the 2nd line.

```
Now is the time  
for all good men to □  
for all good men to
```

<ESC><INS> (<CTRL> U for non-PC) deletes all blanks spaces to the next character. This command is used by placing the cursor at the spot one would like an open space to be closed to. Then give the command and the space is closed.

EXAMPLE: The unwanted space in,
"Rose is a □ lovely baby!"
is closed by giving the command while the cursor is where shown.

ERASING COMMANDS:

<ESC> (<CTRL> Y for non-PC) erases all characters from the cursor to the end of the line. The command is used by placing the cursor at the beginning of the line segment one wishes to erase. Then give the command and the remainder of the line is erased.

EXAMPLE: Everything after "dog." on the line below will be erased by giving the command while the cursor lies where shown:

"The quick brown fox jumped over the lazy dog.□ and he

<BACK> (for non-PC.) The "Delete Key" erases text and replaces it with blank spaces. The cursor, of course, moves from right to left.

SPACEBAR The spacebar erases text and replaces it with blank spaces, too. The cursor, this time, moves from left to right.

COMMANDS TO ADD TEXT:

Overtyping replaces old text with new. Overtyping means simply that you type "on top of" previously typed material, erasing old text as you go. Move the cursor to the point you want to begin overtotyping and begin.

<INS> (<CTRL> W for non-PC) inserts a character space. This command is used by moving the cursor to the spot you want to insert a blank space. Then give the command: A blank space will now appear where the cursor is, and all of the text to the right of the cursor will have shifted one position to the right.

EXAMPLE: "Rose isa lovely baby!" is corrected by pressing <INS> with the cursor over the "a" in "isa".

<F9> (<CTRL> R for non-PC) inserts a blank line just above the line the cursor is on. The text below the new blank line shifts down one line.

EXAMPLE: With the cursor as shown, a blank line will be put between the two lines.

Now is the time
come to the aid □

BREAKING AND JOINING LINES:

<ESC> <F9> (<ESC> > for non-PC) breaks a line--that is to say, it moves everything on a line, from the cursor onward, down to a new line below. Lines that were below the line you "broke" will also be shifted downward, to make room.

EXAMPLE:

"Now is the time □ for all good men . . ." becomes

"Now is the time
for all good men . . ."

<ESC><F10> (<ESC> < for non-PC) joins lines--that is to say, it brings the line below up to the present line, and puts the whole of the line brought-up after the cursor. By the way, if there was text after the cursor (i.e., in the place the line below will be brought-up to), that text will be erased.

EXAMPLE: "Now is the time for □
all good men . . ."

becomes: "Now is the time for all good men . . ."

by using the command when the cursor is where shown.

INSERT MODE: INSERTING TEXT INTO A LINE

<ESC> I allows you to insert text into a line. Touching <ESC> I, in other words, shuts off T/Maker's natural inclination to overtype.

This command works a bit differently from the other editing commands.

<ESC> I is a "toggle option". That means, you use the same command to TURN IT ON and to TURN IT OFF.

Suppose that you want to add the words "very, very" just before the word "nice" to the sentence below.

EXAMPLE: "That was a nice dinner!"

To insert text:

1. Move the cursor to the "n" in "nice."
2. Touch <ESC> I.
3. Type in "very, very".
4. Touch <ESC> I again (to take you out of INSERT MODE and back to normal).

Touching <ESC> I the first time puts you into INSERT MODE. This lets you write new text without erasing old text. It just pushes the old text along to the right.

Touching <ESC> I the second time (after inserting text) takes you out of INSERT MODE and puts you back into normal overtyping.

FINDING AND REPLACING STRINGS

T/Maker provides separate commands for finding and for replacing strings. (You can find or replace both when inside or outside the editor. This subsection, however, describes the inside-the-editor commands, only--see the "List Processing" for the Find and Replace commands which can be used with the Universal Prompt.)

FINDING STRINGS:

This command moves the cursor to the leading character of a prescribed string of text.

1. Move the cursor to a spot above the text area to be searched. (The cursor's location when a search is asked marks the point of departure for the search; text above the cursor will not be searched.)
2. Touch <F5> (<ESC> * for non-PC)--this brings a "FIND WHAT STRING?" prompt to the screen.
3. Answer with the string sought: "_____ <RET>"
4. The cursor now appears at the leading character of the first occurrence of the string.
5. Touch <F6> (<ESC> & for non-PC)--this moves the cursor to the string's next occurrence. Repeat this command for successive occurrences. (When no further occurrences exist, a beep sounds.)

REPLACING STRINGS:

This command replaces a string with a new string.

1. Move cursor above search area (see pt. 1 above).
2. Touch <ESC> <F5> (<ESC> * for non-PC)--this brings a "REPLACE WHAT STRING?" prompt to the screen.
3. Answer with the string to be replaced: "_____ <RET>"
4. This brings a "WITH WHAT STRING?" prompt to the screen. Answer with the string to be substituted: "_____ <RET>"

All string occurrences are replaced. The cursor is returned to the point at which the replacement command was invoked.

Tip: Both strings (i.e., the one to be replaced and the one replacing it) can be given in one go to the "REPLACE WHAT STRING?" prompt by separating the two strings from each other by a space. For example,

```
REPLACE WHAT STRING? car automobile <RET>
```

In this way, a mistake in the first string can be noticed and corrected even when in the middle of typing the second string.

To back out of this prompt, replace a string by itself.

```
REPLACE WHAT STRING? car car <RET>
```

WHAT IS A STRING?

A "string" is simply a specific series of characters--the string may be a word, series of words, a number, a series of numbers or a mixed series of words and numbers.

There is a rule for forming strings:

When a string contains a blank or begins with a single-type or double-type quotation mark, the string must be bounded by quotation marks.

Additionally--for strings needing bounding quotation marks--the quotation marks used for bounding must not be present within the string. That is, if doubles are in the string, then the bounding should be done with singles, and vice versa.

EXAMPLES OF CORRECTLY FORMED STRINGS

The String	As Specified for Searching or Replacing
apple	apple
apples and oranges	"apples and oranges"
Joe's	Joe's
at Joe's	"at Joe's"
"He	'"He'

TABS

T/Maker's TAB key works in the conventional way: touch it, and the cursor moves to the next tab setting. For IBM, the <TAB> key is the one with arrows pointing in both directions.

Setting and clearing tabs is taken care of by four commands:

<ESC> C <TAB>.....CLEARS ALL TABS

<ESC> <TAB>.....SETS A TAB

<ESC> <ESC> <TAB>.....CLEARS A TAB

<ESC> S <TAB>.....SETS ALL TABS

Setting or clearing a tab is done by giving the required command while the cursor sits at the spot the tab is to be placed or removed. Clearing all tabs is a command that can be given at any place in a document.

The command to "Set All Tabs" sets a number of tabs (as many as you like) in a single stroke. This is done by placing the cursor anywhere on a line of text with characters, words, or numbers placed at spots tabs are desired. For example, giving the "Sets All Tabs" command while the cursor lay on the line below would set tabs at the B in Bob, the L in Larry, the S in Sue, and at the 3 in 3,456.

EXAMPLE LINE:

Bob	Larry	Sue	3,456
-----	-------	-----	-------

Note: By default, T/Maker provides you with a series of active tab settings--at positions 8, 30, and every 10th position thereafter. These, as it happens, are particularly useful tabs for many tables.

Also, tab settings can be saved with a file so that when the file is gotten, the tab settings are automatically restored. (See the SAVE, TABS, and NOTABS in the "File Management And Other Commands" section.)

MOVING LINES

T/Maker provides a number of commands for moving or temporarily storing lines of text, while remaining in the editor. Approximately 40 lines can be stored or moved in one go.

Moving and short-storage commands involve the use of the T/Maker's BUFFER, which might be thought of as a handy little memory attached directly to the editor.

Moving lines is done by three steps: first, the cursor is placed on the line to be moved, and the line is "moved" or "copied" into the buffer; next the cursor is moved to the lines' new location; and, finally, the buffer is made to dump out the lines, into their new site. (Old text below accommodatingly moves down to make room.)

Four commands control the buffer:

PC <u>KEYSTROKE</u>	NON-PC <u>KEYSTROKE</u>
<F8> .. <CTRL> G.....MOVES A LINE INTO THE BUFFER	
<F7> .. <CTRL> F.....COPIES A LINE INTO THE BUFFER	
<ESC> <F8> .. <CTRL> C.....CLEAR THE BUFFER	
<ESC> <F7> .. <CTRL> D.....INSERTS THE BUFFER'S CONTENTS	

SOME NOTES ON THE BUFFER:

MOVING a line into the buffer deletes that line from the text and places it into the buffer. COPYING a line into the buffer, on the other hand, leaves that line intact in the text but still places a copy of it into the buffer. In either case, the line in question is available in the buffer.

CLEARING the buffer deletes the buffer's content but does NOT place those contents into the text. The buffer is merely emptied. Clearing the buffer is a good practice before using the buffer for a new task.

INSERTING the buffer's contents, of course, puts the lines stored in the buffer into the text. Inserting the buffer, however, does NOT empty or clean it: You can insert the same buffer over and over again into text.

You may move or copy lines into the buffer in any order--they will later come out of the buffer in the order they went in: That is, the first line to go into the buffer becomes the top line of the text inserted from the buffer into the text. By the way, you can move or copy blank lines into the buffer, too, to maintain the text's original appearance.

KEYSTROKE MACRO

Keystroke macro refers to a set of three steps that allow T/Maker users to "can" a series of keystrokes. Like canning tomatoes, this means you preserve a series of keystrokes for use later on--usually, in order to perform a repetitive and otherwise time-consuming task.

FIRST STEP:

<F3> (<CTRL> B for non-PC) .. BEGINS THE MACRO

From the time this command is given, T/Maker silently begins to register and remember in order every key touched thereafter, until the command to stop this remembering is given.

SECOND STEP:

<F4> (<CTRL> X for non-PC) .. STOPS THE MACRO

This command tells T/Maker that the series of keystrokes to be remembered has been completed.

The Keystroke Macro you have now created is ready for use. All that need be done to use it once is to touch **<F4>** (or **<CTRL> X**) again.

THIRD STEP:

<F4> (or <CTRL> X) (now) EXECUTES THE MACRO

Each time **<F4>** is touched thereafter, the Macro is executed one more time. In other words, the series of keystrokes placed into the Macro is automatically repeated.

HOW TO USE A KEYSTROKE MACRO:

AN EXAMPLE:

Suppose that you had typed the list shown below:

COMING TO THE PARTY:

Don and Ellen
Hilda and Richard
Charles
Denise
Willie
George and Joan
Caroline
Fei-Peng
The Reikens
David R.
Steve and Lena

Suppose that after looking at the list, you decided it would look nicer if the names were aligned a little more to the left, say three positions.

NOT USING A KEYSTROKE MACRO:

There are, in fact, several ways to shift this column of names thus. But suppose that you decided to do the job by moving the cursor to the left margin of the "Don and Ellen" line and then touch the character-delete command () three times, and then hit a carriage return and repeat the process until all the guests were moved over.

This looks like a job for a Keystroke Macro!

USING A KEYSTROKE MACRO:

Move the cursor to the left margin of the "Don and Ellen" line. Now, touch <F3> (beginning T/Maker's Macro memory), touch the character-delete command three times, return the carriage, and touch <F4> (ending the Macro series).

Now, all you need do is touch <F4> to shift each name over.

OTHER MACRO COMMANDS

Two other commands are provided in the Keystroke Macro group:

<ESC> <F3> (<CTRL> N for non-PC) HOMES THE MACRO

This command is essentially a memory helper. After touching **<F3>**, you will recall, T/Maker begins to register and remember subsequent keystrokes. Sometimes, these keystrokes may take the cursor far away from the point where the Macro began. This is important, because Macros almost always form a "loop" of activity that is best begun again IN THE POSITION that the Macro was first started.

Thus, this command is provided as a quick means for returning to the column position in which the macro started but on the current line.

<ESC> <F4> (<CTRL> Z for non-PC) can also be used in place of **<F4> (<CTRL> X for non-PC)** to end the definition of the macro. It has another use, as well, if it is used after the macro has already been ended:

<ESC> <F4> (<CTRL> Z) CAUSES THE REPETITION PROMPT

It produces the prompt below.

"HOW MANY TIMES?"

This prompt is then answered with a number followed by a carriage return. The keystroke macro will be repeated the requested number of times automatically.

REDRAWING THE SCREEN

This command redraws the screen:

<ESC> <F2> (<ESC> # for non-PC).....REDRAWS THE SCREEN

Probably the most common use for this command is to remove the image of the INFO lines from the screen (discussed in next subsection).

More generally, the "Redraw the Screen" command restores the screen to an accurate representation of the working file's live contents. Such restoration can be useful in a situation where a communication between the terminal and the computer has been garbled.

THE INFO COMMAND:

T/Maker provides a convenient command for getting a variety of information about the working file. This is done simply by touching <F2>.

<F2> (<ESC> ? for non-PC).....INFORMATION (OR INFO)

When you touch these keys, three lines of information immediately appear at the top of the screen. They look something like this:

```
NAME: B:JONES.TAB LINES: 93 FREE: 11804 LINE: 18/1 MODE: M/I
.....1.....2.....3.....4.....5.....6.....
12345678901234567890123456789012345678901234567890123456
```

These are called the INFO lines.

These lines will appear to have replaced whatever lines originally lay at the top of the screen. But this is only an illusion--in fact the INFO does not appear in the working file at all, and the original lines are still there intact, though temporarily out of view.

The lower two INFO lines provide a ruler for the screen. This can be handy for setting tabs and other editorial needs. The ruler is probably handiest for specifying the boundary positions for data columns, particularly when using such commands as "Tally" and "Sort" (see the List Processing Section).

THE TOP INFO LINE'S CONTENTS:

The top INFO line reports a total of six pieces of information:

NAME: B:JONES.TAB LINES: 93 FREE: 11804 LINE: 18/1 MODE: M/I

READING THE TOP INFO LINE:

Line Segment	Content
1. "NAME: B:JONES.TAB"	File name and location: The working file is "JONES.TAB" on the B disk drive.
2. "LINES: 93"	File length: The file is 93 lines long.
3. "FREE: 11804"	Free characters remaining in the file: 11,804 remaining.
4. "LINE: 18/1"	The cursor's location when INFO was asked: line 18, position 1.
5. "MODE: M/I"	This tells two things: a) The character before the slash indicates the Carriage Return Mode presently in force (M for Manual, T for Table, and D for Drafting Mode): Manual. b) The character after the slash indicates whether the editor is presently in the Insert Mode or in the normal, overtyping mode (see "Revising Text"). "/I" indicates that the Insert Mode is presently on. When the editor is in the normal mode no slash or letter indication is presented.

MOVING COLUMNS:

T/Maker provides a simple method for moving columns of data from one place to another. The method involves four commands:

```
<ESC> ( ..... tells T/Maker the  
LEFTMOST limit of the  
columns to be moved  
  
<ESC> ) ..... tells T/Maker the  
RIGHTMOST limit of the  
columns to be moved  
  
<CTRL> G ( <CTRL> V for non-PC) .. MOVES designated columns  
to the cursor's present  
site  
  
<CTRL> F ( <CTRL> S for non-PC) .. COPIES designated columns  
to the cursor's present  
site
```

MOVING COLUMNS commands are used as follows:

STEP 1:

Locate the cursor at the leftmost position of the column-group to be moved; touch <ESC> (.

STEP 2:

Locate the cursor at the rightmost position of the column-group to be moved; touch <ESC>) .

STEP 3:

Locate the cursor at the desired site for the column-group just designated.

STEP 4:

a) If it is desired that the column is DELETED from the home location and appears at the desired site only, touch <CTRL> G (<CTRL> V for non-PC).

b) If it is desired that the column REMAINS at the home location, and A COPY of the column appears at the desired site, touch <CTRL> S (<CTRL> F for non-PC).

STEP 5:

Touch the same keys again to repeat the column moves in successive lines.

Notes: This group of commands moves columns a line at a time--so it is necessary to repeat the final command to move the column on successive lines. This is made easier by an automatic advance to the next line after the command is given. (Tip: to make the process faster, the column move can be made a part of a Keystroke Macro and executed a number of times in one go.)

The column moving commands may of course be useful in editing text--for example, in order to change the location of a phrase in a given line.

One rather technical point about this command concerns the destination site for the column. In fact, T/Maker does not move a column group to the exact position indicated by the cursor. Instead, the cursor location informs T/Maker of the ORDER in which the columns should appear. Hence, the instruction given by the final column-moving is more like saying "the column about to be moved should be placed between these two positions" rather than, "the column to be moved should be placed at positions 35-40".

FRAME MODE:

FRAME MODE separates the screen into two portions by drawing one of three types of bars on the screen:

1. A horizontal bar separating the top from the bottom of the screen (the bottom is the "inside" of the frame).
2. A vertical bar separating the left side from the right side of the screen (the right side is the "inside" of the frame).
3. An L-shaped bar separating a top and left-side portion of the screen from the lower right-hand corner ("inside").

<ESC> F puts the frame on the screen, or takes it off if it has been on

Mode	To Enter	Properties
1	Put the cursor over the leftmost position of the first line to appear under the barrier. Type (ESC) F.	When the screen is moved, the lines shown above the barrier remain. Thus, the "working area" remains static as well. This means that action type commands like Compute can be used while working in the middle of a table.
2	Put the cursor on the top line over the first character to appear to the right of the barrier. Type (ESC) F.	When the screen is moved, the positions shown to the left of the barrier of each line will remain.
3	Put the cursor on top of the first line to appear beneath the horizontal barrier at the location of the first character to appear to the right of the vertical barrier.	Both properties above.

Notes: When entering Frame Mode, there is no rule that says the top line on the screen must be the first line in the file or that the leftmost position displayed on the screen must be the first position in the line. Also, you are not obliged to leave Frame Mode when you quit the Editor. In Frame Mode, you can only edit what is inside the frame. However, you can arrange by moving the screen left, right, up, and down to have parts of the frame inside the frame as well. When changes are made in this manner, the change will not be reflected in the frame itself until the Editor has the occasion to redraw the part of the frame that has been changed. You can force the Editor to redraw it, of course, by using <ESC> #.

Some movements of the cursor work differently though remaining logical. For example, if you Move to the First Line, the top line in the interior of the frame will be the first line after the frame rather than the first line in the file.

MODE 1

Name	Phone Number	Age
<hr/>		
Bob Johnson	(415) 623-0001	32
Karen Stevens	(812) 444-1945	29
George Barnes	(212) 534-1286	48

MODE 2

Jones	1234 West Main Street
McDonald	842 First Street
Barnes	1289 University Avenue

MODE 3

Item	Number Sold	In Inventory
	99,999	99,999
<hr/>	<hr/>	<hr/>
Jars	313	450
Cans	112	612
Seals	99	244
<hr/>	<hr/>	<hr/>
Total	846	3,421
<hr/>	<hr/>	<hr/>

THE ALIGN COMMAND

The Align command aligns text according to the user's instructions. This is the T/Maker command that makes text right-justified or right-ragged, and centers headings or other text, thus shaping the text into its final appearance.

To align text, the user types "aligning wedges" on to an otherwise BLANK LINE that PRECEDES or is ABOVE the text desired to be aligned.

This is done while remaining in the editor. The left-side wedges define the left margin* and the right-side wedges define the right margin for all text that follows them--until another pair of wedges appear, superceding the first pair.

Aligning wedges look like this:

<<

>>

(Wedges come in different varieties, each having a different aligning effect. See facing page.)

Nothing happens when the wedges are first placed into the text. This is because the wedges have not yet been "enforced". When one is ready to enforce them--that is, put into effect the wedges' aligning instructions--the following steps should be taken:

1. Locate the cursor above the text area to be aligned.
2. Quit the editor.
3. 1/1 WHAT NEXT? ALIGN

In a moment, the aligned text and the Universal Prompt will appear on the screen. The user may now resume editing the text, or print the document, or whatever.

* For straight text with no embedded tables, experience suggests that setting the left margin in position three is a good choice. Why? Print Design Commands stand out from the text; T/Maker's first default tab setting is in position 8 (i.e., 5 spaces from the left margin); a Non-Aligning Line, if needed, can only print text in positions three or higher; and, there is room to overhang the bold and underline toggle characters to the left of the text.

T/MAKER'S BASIC ALIGNING OPTIONS:

ALIGNING WEDGES		ASSOCIATED ALIGNING EFFECT
<<<	>>	Block Text, Right-justified
<<<	->>	Block Text, Right-ragged
<<-	>>	Centered Text (or Headings)
<<	>>	Indented or Numbered Paragraphs,* Right-justified
<<	->>	Indented or Numbered Paragraphs,* Right-ragged
>><<		Stop Aligning
<<or..... >>		Single-side Changes**
<<m	>>	Tagged Wedges**
<<=m		Recalled Wedges**

[i.e., no wedges]
No Aligning

Notes: Aligning wedges may be placed throughout a text. Each pair of left and right wedges prescribes the margins for the text appearing below them. These margins stay in effect until a new pair of wedges comes along.

Ordinarily the line on which the margin wedges appear will not print (see "Wedges" in "The Print Command", for the exception). This means that if one needs to insert a blank line into text in order to provide a line for wedges, that inserted line will not print. Thus, the line-spacing of the original text will not be affected.

* See "Indenting or Numbering Paragraphs"

** See "Aligning Shortcuts"

BLOCK TEXT

Block text is the simplest aligning option. Block text is gotten by using three left-side wedges and two right-side wedges, as below:

<<<

>>

Block text has a straight-line margin on the left side. This text has a straight-line right margin as well. That is because the right-side wedge, just above, is the command for RIGHT-JUSTIFIED text. So, this particular paragraph is an example of RIGHT-JUSTIFIED BLOCK TEXT. Right-ragged text is gotten by inserting a hyphen just inside the right-side wedges. Like this:

<<<

->>

Now the right-side margin has the irregular appearance of conventionally-typed text. This paragraph is an example of RIGHT-RAGGED BLOCK TEXT. Right-ragged and right-justified text also differ in their handling of blank spaces: In the right-justified case, extra blank spaces are supplied to the line so that the right-side margin comes out straight; in right-ragged text, on the other hand, no such extra blank spaces are supplied.

<<<

>>

So that the Align command does not run text together into one big, blocked paragraph, the user must inform T/Maker when a new paragraph is begun. The simplest way to do this is to leave a blank line between paragraphs--THE ALIGN COMMAND WILL NOT STRING TEXT ACROSS A BLANK LINE. There are other ways to signal the beginning of a new paragraph, too, though, and some do not require that paragraphs are printed with interspersed blank lines (see Paragraph Demarcation).

Of course, it is not necessary to supply aligning wedges before each new paragraph--as above. One set of wedges stays in effect until a new set comes along, superceding it.

Note: a single line of text that does not reach the right margin, (e.g., the line above) will not be aligned. This rule is handy for excluding paragraph headings from the Align command.

PARAGRAPH DEMARCATI

The demarcation of paragraphs can be done in a number of ways.

In regard to demarcation, THE CENTRAL NOTION TO BE GRASPED IS THAT THERE EXIST A NUMBER OF BARRIERS ACROSS WHICH THE ALIGN COMMAND WILL NOT STRING TOGETHER TEXT. Thus, these barriers, wherever they appear, will mark the beginnings of new paragraphs.

ALIGN BARRIERS

1. A BLANK LINE: ALIGN will not string text together across a blank line; thus a new paragraph is begun wherever a blank line is put into the text. The blank line is also "printed" by the printer, which puts a blank line between paragraphs in the final output.
2. A PERIOD (.) IN COL. 1: ALIGN will not string text together across a line that begins with a period. Lines that begin with a period are not printed by the printer, either.* Therefore, periods in the first column provide a handy way to carve out paragraphs without necessarily obliging the printed text to include blank lines between the paragraphs.

Incidentally, lines with periods in the first column are a part of T/Maker's "design commands" (see the "Print Command").

3. ALIGNING WEDGES: A line with wedges provides both a paragraph barrier and does not print.

4. DASHES OR UNDERSCORE CHARACTERS: A line with three or more dashes or underscore characters in succession will not be aligned. Such lines are typically used for decorative purposes.

* An exception to this rule is a line beginning with a period in the first position and a comma in the second position (i.e. a Non-Aligning Line--see the Print command). The period and comma are removed, but the line is printed.

INDENTING OR NUMBERING PARAGRAPHS

The Align command provides a convenient way to indent or to number block paragraphs.

The shared characteristic of indented paragraphs and numbered paragraphs is that their FIRST lines do not align uniformly with subsequent lines.

The ALIGN command provides wedges that align the second and following lines of text, but do nothing to the positioning of the beginning of the first line of each paragraph.

For indented or numbered paragraphs, use two left-side and two right-side wedges, like these:

<<

>>

The first line of this text is free to start where it likes. Subsequent lines of text, though, will be aligned just the same way they would have been by the "Block Text" wedges.

<<

>>

1. Thus, this method provides a convenient way to construct numbered paragraphs. By placing the left-side wedge where the indented text's left margin should appear, text will be aligned in the desired fashion.

<<

->>

2. This method of alignment also provides right-ragged text, simply by adding a hyphen just before the right-side wedges.

This aligning option follows the same paragraph demarcation conventions as Block Text (see "Paragraph Demarcation").

CENTERING

The centering of headings or other text is done by adding a hyphen to the left-side wedges, like this:

<<-

>>

Single
words, short headings,
and longer strings of words will be centered around
the midpoint between
these wedges.

Because centering is often a means for designing
artful
text, the centering wedges do not break lines or alter
the spacing that the author supplies.

<<-

>>

For that reason, centered lines may
exceed the margin boundaries set out by the wedges.
In other words, the centering wedges will not send a word
from one line to another.

STOP ALIGNING

There are a variety of circumstances in which it is desirable to stop the aligning command.

This is done by using reversed wedges (i.e., left-side wedges have been placed on the right side, and right-side wedges have been placed on the left side):

>> <<

All text appearing below these wedges will be
overlooked
and not aligned by the Align command.
Aligning will begin again
only when conventionally-ordered wedges are put in the text.

NOTES ON STOPPING ALIGNING:

There are other ways to thwart the text-aligning, too. One way is, of course, to install the first pair of aligning wedges into the text BELOW other text one wants to preserve unaligned. A typical circumstance for this tactic is the business letter address block. By installing wedges in the manner described below, one will avoid aligning the address block:

Mr. William Winocur
4152 Easy Street
Berkeley, California 94709

Dear Mr. Winocur:

<<

>>

Thank you very much for your recent inquiry. I am sorry to report that we no longer carry the shirt you described. Nevertheless, it may be possible . . .

With the wedges placed below the address, the address, of course will not be aligned, and thus rendered into a single line of text.

Another means for thwarting the Align command is simply to invoke the command when the Top Line on the Screen is below text which should not be aligned.

Naturally, text which is not in the "working area" will not be aligned.

Finally, a special line called a "Non-Aligning Line" can be used. It begins with a period in the first position and a comma in the second position. The line will never be aligned even if preceded by aligning wedges. It will be printed, however, but without the period and the comma.

ALIGNING SHORTCUTS

SINGLE-SIDE CHANGES:

Should the user desire to change only one margin--say, change the right-side margin without changing the left--T/Maker provides a shortcut: Namely, it is necessary to position only one wedge-side on the side to be changed. This method is illustrated with the two paragraphs below:

<<

>>

This paragraph begins with the right margin set at sixty-three columns. In order to bring that margin in, and change to a right-ragged format, it is necessary to move only the right-side wedges.

->>

Notice that this paragraph preserves the setting of the left margin from the previous paragraph.

TAGGED AND RECALLED MARGINS:

Sometimes it may prove useful to identify or tag sets of margin-wedges for future use. A good example of this occurs in the typing of a report containing quoted material, material that is placed between narrower margins than the regular, unquoted text.

In this case, it is handy to prepare and "tag" two sets of margin-wedges, one for wide text (named "w") and the other for narrower text (named "n"). This single-letter name is placed on the line with the wedges, like this:

<<n

>>

<<w

>>

The advantage bestowed by using these "Tagged Wedges" is that either the "n" or the "w" margins can be recalled into use by typing simply a "<<=n" or a "<<=w". For example, let us commence the next line within the narrow margins.

<<=n

Now the text will be aligned within the "n" margins. (Of course, it might be kept in mind, too, that because two left-side wedges are in use, the left side of the first line of each paragraph is not aligned--see "Indenting or Numbering Paragraphs". Therefore, it is necessary that the beginning point of the first line is controlled. Setting a couple of tabs for that purpose is sufficient.)

<<=w

Now we have returned to wide margins. Perhaps the main benefit of these "Recalled Wedges" is that a particular set of margins can be defined once and only once at the beginning of a file. If a new type of alignment is required, only "Tagged Wedges" need be changed to reformat all the text.

<<=n

(These wedges with equal signs and letter names, which recall into effect the earlier margin settings, are called "Recalled Wedges".)

<<=w

Only a one-letter name can be given tagged wedges. Any type of wedges may be tagged (block text, centered, etc.) and recalled.

Recalled wedges always begin with "<<=" even if the left margin setting being recalled was "<<-" or "<<<".

EXACT SPACING WITHIN ALIGNED TEXT

The Align command enforces a number of rules to achieve aligned text. In some cases, these rules may produce an undesired result. Typically, this happens when an exact spacing is required between two words.

For example, Align places two spaces after a period when text is aligned. Thus "Mr. Jones" will have at least two, and perhaps more, spaces between "Mr." and "Jones". Suppose, one is searching for perfection. How can "Mr." be put one space from "Jones"?

First Solution: Select a character which is not used in the text file and place it between "Mr." and "Jones" like this-- "Mr.%Jones". Just before printing, replace the character with a single space.

Second Solution: Use T/Modify Option 2 (see Customization Notes) to create a "High-Bit Character," say the "@" sign. Now, when typing "Mr. Jones" put a high-bit space between "Mr." and "Jones" instead of a regular space by touching the "@" before the space. This will print on nearly all printers as a regular space. Align will regard it, however, as a non-blank character.

Third Solution: Pick a character on your keyboard that you never use in a printed file, say "^". Now use T/Modify Option 3 (see Customization Notes) to request that printing be done with a "Printer Translation Table." This involves answering "yes" to one question.

Create a file named "PRINT.UTL" on the same disk as the T/Maker programs. Type in the following line leaving some space between the "=^", "=", and the zero as shown. Save the file.

=^ = 0

From then on, a "^" on the screen would be printed as a space on the printer. Such a non-blank blank could be used in any case where text is to be aligned but exact spacing is necessary within parts of the text.

THE PRINT COMMAND

The Print command superintends both the manufacture of the printed document and many features of the document's appearance.

Regarding document manufacture, this command provides for such options as: designating the files to be printed, whether printing should be done "to the printer" or "on the screen", or whether printing should run continuously or with pauses at the start of each new page (in order, say, to load paper).

Regarding document appearance, this command provides for the control of such options as:

- single or double spaced text
- boldfaced text
- underscoring
- repeated headings or page-endings
- automatic page numbering
- footnotes
- blocking text (that is to say, obliging a portion of text to appear on a single page)

As we will see, the Print command does other things as well.

TO PRINT THE WORKING FILE

One may print out the working file (that is, the file on the screen) at any time. The procedure:

FIRST:

Move the cursor to the beginning of the material to be printed. (The Print command prints the working area).

SECOND:

Quit the editor.

THIRD:

1/1 WHAT NEXT? PRINT IT

FOURTH:

The "PRINTER'S PROMPT" will now appear. It looks like this:

"NEXT PAGE 1 (YES.SCREEN.NO.GO.QUIT)?"

Type "y" to print the first page (the prompt will reappear for the next page); or type "g" to print all pages without further prompting.

No carriage return is required after these responses.

If it is necessary to stop printing after starting it, type:

<CTRL> Q (CP/M) or <ESC> (Other)

THE PRINTER'S PROMPT

A request to print brings on the appearance of the PRINTER'S PROMPT:

NEXT PAGE 1 (YES,SCREEN,NO GO.QUIT)?

With this prompt, T/Maker is asking the user what to do with the page that is currently first in the printing queue. The "NEXT PAGE 1" part of the prompt tells the page upcoming. The five words within the parentheses cryptically represent the following options:

YES -- for YES, print the next page to the printer
SCREEN -- for SCREEN, print the page to the screen
NO -- for NO, do not print this page, but ask me about the next page
GO -- for GO ahead, print the file continuously without checking with me for approval to print each page
QUIT -- for QUIT the printer--return me to the "WHAT NEXT?" prompt

A response is given by typing the first letter of the desired option.* A carriage return is not necessary.

* Some options of the Print command may cause the PRINTER'S PROMPT to be answered automatically for some or all pages (see next subsection).

The PRINTER'S PROMPT may not be answered in advance by the user directly like other T/Maker prompts. The following is incorrect and will produce an error.

1/1 WHAT NEXT? PRINT IT Y Y N N S Y

PRINTING TO THE SCREEN

Printing to the screen is one of the options offered by the Printer's Prompt. This option provides a good way to preview the contents and layout of a document, without having to commit it first to paper.

Just as the printer does, printing to the screen produces a running display of the file that comes to a halt at the end of each page.

One can halt the screen-printing before the end of a page by touching the spacebar (or any key). Touching a key again will start screen-printing again.

At each page's end the Printer's Prompt comes back to the screen, providing the user the option to continue the screening of the text or take one of the other choices for the new page.

The speed of the running display can be adjusted to the user's liking--see "Delay 15" in the next subsection.

One of the best utilities for the print-to-screen feature is checking the pagination of a document.

Like the printer, screen-printing can be terminated in progress or "killed" by touching the kill character, <CTRL> Q.

THE PRINT COMMAND, MORE GENERALLY

One may begin the process of printing a file by answering "PRINT <RET>" to T/Maker's "WHAT NEXT?" prompt.

T/Maker now responds with a question:

"WHAT OPTION OR FILE (OR 'IT')?

T/Maker is asking both WHAT is to be printed and if there are any special instructions concerning its printing.

WHAT SHOULD BE PRINTED

On the question of WHAT should be printed, there are two sorts of legitimate responses:

IT

or,

FILE-NAME

The "it" response is an instruction to print the working area. The "file-name" response tells T/Maker to print a file from disk. The file to be printed from disk should be sought by the same name that would be used to GET it.

SPECIAL INSTRUCTIONS

Five kinds of special instructions may be requested. These are associated with five keywords: NONSTOP, DELAY, FROM, TO, and NUMBER.

Special instructions must always be given before responding with the designation of the file to be printed. (i.e., "it" or the "file-name"). Responding to the prompt with a special instruction will cause T/Maker to accept the instruction and present the prompt again. For example, to request NONSTOP printing of the working area, the series of questions and answers would be:

WHAT NEXT? PRINT

WHAT OPTION OR FILE (OR 'IT')? NONSTOP

WHAT OPTION OR FILE (OR 'IT')? IT

LEGITIMATE RESPONSES TO:
"WHAT OPTION OR FILE (OR 'IT')?"

NONSTOP: When this option is requested two consequences result. The file to be printed is sent directly to the printer, and the printer prints the file continuously, page after page.

Ordinarily, there are two kinds of decision points following the request to print. The user's attention is required for both--the user must respond both to the "PRINTER'S PROMPT" and to successive requests to approve or disapprove the printing of individual pages as they come forth in the printing queue.

Both these decision points are shortcut when the user requests "NONSTOP" printing--The material to be printed goes directly to the printer, and it then prints nonstop to completion. By stacking NONSTOP print requests (like this: "PRINT NONSTOP 'file name' PRINT NONSTOP 'file name' PRINT NONSTOP 'file name' <RET>" several files may be printed in sequence without the user's presence required. (Of course, continuous printing requires the hardware for continuously fed paper, or users of superior dexterity.)

DELAY 15: This option pertains only to printing to the screen. It sets the rate at which the printer writes characters across the screen. Zero is the default rate, which is quite fast; 20 slows the rate down considerably. The user may find a value to his liking. Once set, the delay factor need not be reset for each printing. T/Maker keeps the same delay factor in effect until superceded by another or until T/Maker is terminated and control returned to CP/M.

FROM 9: This option permits the user to skip the printer to the designated page. If, for example, one wanted to reprint only page 9 of a nine page report, this option more speedily moves the printer to that page. Once the page is reached the PRINTER'S PROMPT appears and should be answered.

TO 8: This option permits the user to stop the printer after the designated page. If, for example, one wanted to reprint only through page eight of a 20 page report, this option when used with NONSTOP eliminates the necessity to answer the PRINTER'S PROMPT for the pages to be printed.

NUMBER 7: This option gives the first page the page number designated. Unless specified, the first page is assumed to be number one. Page numbers run continuously thereafter, starting with the number designated. If "0" is assigned to the first page, the page number will not be incremented with each new page (see the .NUMBER design command for inscribing the page number of the first page into the document).

STRINGING AND STACKING PRINT COMMANDS

This discussion of print commands has walked through the stages of print requests.

In fact, the expression of these commands, and the speed of their implementation, can be greatly increased both by "stringing" and "stacking" them together.

For example, the following string of commands can be given to the "WHAT NEXT?" prompt:

```
1/1 WHAT NEXT? PRINT NONSTOP FILE-NAME
```

This command, of course, will start the printer going right away, without a pause at the PRINTER'S PROMPT.

Similarly, the following string will move one from the "WHAT NEXT?" prompt to the PRINTER'S PROMPT for the 19th page.

```
1/1 WHAT NEXT? PRINT FROM 19 IT
```

The order of presentation of these commands is always as follows: first, PRINT; next, the special instruction (if any); and, finally, the file designation.

Several special instruction commands can be included in a single printing session. For example, the following directs that the printer print nonstop from page 12 through page 15 of the document:

```
1/1 WHAT NEXT? PRINT FROM 12 TO 15 NONSTOP IT
```

Notice, once more, that the special-instruction commands are always sandwiched between the "PRINT" and the file-designation ("IT", in the case above).

"STACKING" refers to joining together print commands for different files, which is also permissible. For example,

```
1/1 WHAT NEXT? PRINT NONSTOP FILE1 PRINT NONSTOP FILE2
```

This command, of course, will print two files sequentially.

DESIGN COMMANDS

Design commands provide the user with a great variety of options for shaping the appearance of the printed document.

All design commands are placed into the document while the user is in the editor.

There are a few general points that might be made about design commands:

1. Design commands are placed on a line and begin in the first position with a period (".").
2. The instruction carried by the command is inscribed in the spaces immediately to the right of the period.*
3. Nothing happens to the document's appearance on the screen when design commands are first placed into the document. Design commands manifest their effect only when the document is being printed, either by the printer or on the screen.
4. Design commands are read by the Print command in a downward fashion. In other words, a given command has effect only on text appearing below it in the document.
5. Valid design commands beginning with a period do not print.*
6. Lines that begin with a period do not take up space, either.* Thus a design command can be interposed between two adjoining lines of text by inserting a line and inscribing the command upon it. When the text is printed, the lines will appear in their adjoined form again.
7. Design commands do, however, provide a barrier to the Align command--which is to say, in general, the Align command will not string text across a line that begins with a period.

These principles provide the ground rules for the use of all of the design commands described in the following pages.

* Keywords in design commands may be abbreviated to their first three letters (e.g., "new" for "newpage"). Also, after the end of the specification of the design command, a comment may usually (there are a few exceptions) be put on the line:

.CONTINUE TEXT.5 THIS IS THE TABLE OF AVERAGES

** The Non-Aligning Line is an exception to this rule.

SINGLE OR DOUBLE-SPACED TEXT

T/Maker's default is single-spaced printing. Double-spaced printing is gotten simply by typing a period and the word double, starting in the first position of an otherwise blank line. Like this:

.DOUBLE

This command is typed into the text while one is in the editor. When the document is printed, all lines below the line holding the ".double" command will be printed double-spaced. (As mentioned, the command itself will not appear in the printed text--though here the command has been made to appear for instructional purposes.)

Text can be returned to single spacing simply by typing ".single", starting in the first position of an otherwise blank line. Like this:

.SINGLE

The ".double" and ".single" commands can be interspersed in order to produce alternating double and single-spaced text. Of course, uniformly single-space text requires no design command, since single-spaced text is T/Maker's default condition.

BLOCKING TEXT

Blocking text means forcing a particular set of lines to appear on one page and one page only. Text is "blocked," for example, in order to insure that a table appears entirely and intact on a single page.*

To begin a block, type ".block", starting in the first position of an otherwise blank line. This line, of course, should be immediately above the body of text desired to be blocked. The command:

.BLOCK

To end the block, type ".end", once more, starting in the first position of a blank line. Of course, the ".end" command should be placed on a line just following the text desired to be blocked.

In a few following paragraphs, the use of .block design commands is shown.

.BLOCK

JOINING HEADINGS AND TEXT

When organizing the layout of a long document, it is frequently advantageous to block a heading with a paragraph or two of the text under it. This insures that the heading will not be stranded with a line or two of text at the bottom of a page.

This approach also tends to generate pages of slightly different length. Thus, when text is added to a given paragraph, the changes tend to spill over only a page or two if they spill over at all.

.END

* Footnotes (see Footnotes) are also "blocked" though forcibly placed at the bottom of a page. In instances where a body of text should be blocked, but need not appear in an exact position relative to text before it and after it, "blocking by footnote" will often lead to less wasted space.

See also "Optional New Page" as another means of achieving blocked text.

.BLOCK

BLANK LINES IN BLOCKS

It is best to leave any blank lines required before a block or after a block, outside of the formal definition of the block. If blanks are included in the start of a block, and the block requires advancing to a new page, then the blank lines will be printed at the top of the new page. If they are ahead of the definition of the block, they would be skipped in such circumstances.

Blank lines at the end of a block may force it to be put on the next page whereas it might have fit on the current page if they had been excluded.

.END

.BLOCK

DESIGN COMMANDS IN BLOCKS

Not all design commands may be put inside a block. For example, it would not make sense to put a design command forcing a skip to a new page in the middle of a block. Thus the block is considered terminating not only by an ".end" design command, but also by any of the following:

.block	.top	.bottom	.footnote
.23	.length	.newpage	.pagesize

.BLOCK

The ".block" command could itself be used as its own delimiter. Placing, for example, a ".block" at the beginning of each paragraph would both terminate the previous block and start the definition of the following block.

.BLOCK

The result would be that no paragraph would be split across two pages.

.END

LIMITS ON THE SIZE OF BLOCKS

A block may not be bigger than about 40 lines of 80 characters. If it is, T/Maker will produce an error message and printing will stop. In such cases, the block will have to be shortened or some other design command might be used to achieve a correct presentation (see "Optional New Page").

FORCED NEW PAGE

This command permits the user to force the printer to begin a new page at some point in the file. This command is good for making sure a new chapter begins on a new page.

The new page is forced by typing ".new" in the first position of a blank line:

.NEW

Lines of text below this command will appear on the next printed page.

Text on the newly begun page will appear as far down from the top line of the new page as there are blank lines between the "FORCED NEW PAGE" command and the first line of text.

OPTIONAL NEW PAGE

It is also possible to have the printer advance to a new page on an optional basis. The decision to advance or not to advance is made on the basis of how many lines are left on the page.

In this case, the ".new" command is followed by a number which represents the minimum number of lines that must be left on the page. If fewer than the specified number are left, a new page will be started.

.NEW 4

In the example above, a new page would be started if less than four lines were left when this command was encountered. This command can often be used as substitute for "blocking."

"WIDOWING" LINES

"WIDOWING" usually refers to a situation in which the first line of a paragraph is stranded by itself on one page and the rest of the paragraph is on the next page. Generally, T/Maker will not print a last line on a page if the previous line was blank.* Thus, for single-spaced text with blank lines separating paragraphs, WIDOWING does not occur.

* The main exception to this rule is when lines of text have been "blocked" together. "Blocking" is stronger than the desire to avoid "widows."

However, consider the situation below.

```
[line 1] The correct command is:  
[line 2]  
[line 3]    "compute clean"
```

Here, it would obviously be preferable to allow "line 3" to be the last line on the page. Assuming, for some reason, that one does not want to block the three lines together. The correct approach to permit the "widow" is to disguise line 2 so that it does not appear to be blank. This can be done by making line 2 a Non-Aligning Line with nothing on it (see Non-Aligning Line).

When double spacing is used and paragraphs are not separated by a blank line in the text file, WIDOWING does occur. This can be avoided by separating paragraphs by the "optional new page" design command below.

.NEW 3

START TEXT ON LINE "X"

As the heading suggests, this design command simply instructs the printer to put the next line of the file on the line specified. The command consists of a period followed by the number of the line at which text is desired next to start. Like this:

.10

This command would initiate the next line of the file on the tenth line from the top of the current page. Note that this is NOT a command to skip ten lines. This command "sees" a fixed page made up of an absolute number of lines, and places the next line of text on the appointed line.

Notes: If you happen to ask for the next line of text to be placed on a line that is above the placement of the command on the current page, this command will skip forward to the next page in order to satisfy the line placement request.

"START THE NEXT" commands are particularly handy in conjunction with "FORCED NEW PAGE" commands. By placing a "START THE NEXT" command on the line below the "FORCED NEW PAGE" command, the text of a new page is made to begin on the appointed line.

"START THE NEXT" commands can be sprinkled throughout the text, and many can appear on the lines of one page, if desired.

PAGE-HEADINGS AND PAGE-ENDINGS (TOP AND BOTTOM ROUTINES)

T/Maker's method for placing repeated page-headings and page-endings is part of a more general capability.

This capability we call "top and bottom routines". Top and bottom routines involve the use of the ".top" and ".bottom" design commands.

TOP ROUTINES:

The ".top" command might be described by the kind of assertion it makes to T/Maker.

The ".top" command says: 'Each line following me, until the ".end" command comes along, must be repeatedly printed at the top of every page after me. In other words, I [the ".top" command] mark the top boundary of a zone of space that is reprinted on every subsequent page; the ".end" command marks the lower boundary of that zone.'

To prepare a heading to be used on every page, first, type a ".top" command starting in the first position of a blank line that precedes all "content-lines" of the document.*

Next, on the next line (or two or three lines down, depending on how far down from the top of the page one desires the heading to appear) type the lines of the heading horizontally positioned as they should appear.

Finally, type the ".end" command a line or two below heading. The number of blank lines between the heading and the ".end" command is the number of blank lines between the printed heading and the first line of text on each page.

* A "content line" line is any line that the printer will print--in general, content lines consist of all lines that do not begin with a period or contain aligning wedges. Note that by this definition, a blank line is a legitimate "content line".

In all, the commands should look something like this in the file:

.TOP

REVISION 8.3

Circulation: B

January 23, 1982

.END

These three commands, thus situated, will produce the heading at the top of each subsequent page.

BOTTOM ROUTINES:

The ".bottom" command makes an assertion similar to the ".top" command--only the location of the zone of repeated text is changed.

The ".bottom" command says: 'Each line following me, until the ".end" command comes along, must be repeatedly printed at the bottom of every page after me. In other words, I [the ".bottom" command] mark the top boundary of a zone of space at the bottom of a page--and in that zone T/Maker should print the text between me and the ".end" command on every subsequent page. The ".end" command marks the lower boundary of that zone.'

Bottom routines, incidentally should not be confused with footnote commands (see "Footnotes").

Bottom commands are useful, obviously, for repeated page-endings. Bottom routines can be placed at any point before the last content line of the first page in a document that is to make use of the bottom routine on all pages. The following is an example of a bottom routine:

.BOTTOM

This project was made possible through the generous support of the Clagett Foundation.

.END

PLACING TOP AND BOTTOM ROUTINES IN A DOCUMENT

The text of a top or bottom routine need not be printed on all pages of a document. Their printing can be controlled by where the routines are placed in the document.

TOP ROUTINES:

A top routine will never be used on any printed pages that include "content-lines" placed in the file ahead of the top routine. For example, consider a document starting like this:

INSTRUCTION MANUAL

.TOP

Instruction Manual (continued)

.END

T/Maker would print the two content lines (i.e., "INSTRUCTION MANUAL" and the blank line) before "discovering" the top routine. At that point, it would be too late to use it for the top of the current page. It would only appear on subsequent pages.

BOTTOM ROUTINES:

A bottom routine will never be printed on a page which is completed before it is discovered. Consider the example below.

[last content line of previous section]

.NEW

.BOT

Copyright 1982 John Doe

.END

In this example, T/Maker would have printed the last content line of the section and started a new page before "discovering" the bottom routine. The routine could only be used on the new page or subsequent pages.

CHANGING TOP AND BOTTOM ROUTINES

Top or bottom routines may be changed within a document by placing a new ".top" or ".bottom" design command and following it by the text of the new routine.

A previously defined top or bottom routine can also be replaced by none at all by leaving no lines of text or space between the ".top" or ".bottom" design command and the corresponding ".end" command like this:

```
.TOP  
.END
```

In the example below, a top routine of no lines is introduced after the last content line of Chapter 11. "CHAPTER 12" will thus head the next new page. Then, too late to be used on this page, a new top routine is introduced.

```
[last content line of Chapter 11]  
.TOP  
.END  
.NEW
```

CHAPTER 12

```
.TOP  
Chapter 12 (continued)  
.END
```

Bottom routines can be manipulated in a similar way though there is more flexibility in placing them. Since one typically wants to change routines with the start of a new page; one has a whole page to get the act together before the bottom of the page arrives.

```
[last content line of Chapter 11]  
.NEW  
.BOTTOM
```

Chapter 12 is sadly still in draft form.
.END

CHAPTER 12

DELAYING THE USE OF TOP AND BOTTOM ROUTINES

It is also possible to delay or control the use of a top or bottom routine by insisting that it lay dormant until a certain page-number is achieved. This is accomplished by placing the start-up page number after the ".top" or ".bottom" design command.

In the example below, the top routine would not be used until page-number two of the document even though the routine might have been placed at the head of the document. The bottom routine would not be used until page-number one. (It is possible to have a page number of zero--see ".number" design command.)

.TOP 2

Instruction Manual (continued)

.END

.BOTTOM 1

Copyright 1982 John Doe

.END

It is not possible to have more than one top and one bottom routine laying dormant, thus the technique below will unfortunately not work.

.TOP 1001

PAGE ONE THOUSAND AND ONE

.END

.TOP 1002

PAGE ONE THOUSAND AND TWO

.END

OTHER FACTS ABOUT TOP AND BOTTOM ROUTINES

Only the most simple design commands may be put in a top or bottom routine. These are:

.. .+ .- .= .. .

Any other design command will end the definition of the top or bottom routine as though a ".end" had been placed ahead of the

inappropriate design command.

Top and bottom routines are always printed single-spaced. (Of course, blank lines can be put in to achieve a double-spaced look).

Top and bottom routines are also subject to size limitations. An error message is generated if a top or bottom routine is too large and printing is stopped. The only recourse is to delete some lines or reduce the length of lines between the ".top" or ".bottom" and the corresponding ".end".

NUMBERING PAGES

Continuous page numbers can be automatically applied to the document by using top and bottom routines. The number sign (#) placed anywhere in a top or bottom routine is replaced by the page number when the text of the routine is printed.

Thus to have centered page numbers at the bottom of every page, the following bottom routine could be placed at the head of the document.

.BOTTOM

#

.END

If the page number were more than nine, T/Maker would insert additional space to the right of the number sign for additional digits. When right-adjusted numbers are desired, more than one number sign may be used as in the example below which would number pages starting with the second page. Note that two birds (headings and page numbers) are killed with one stone (the top routine).

.TOP 2

Report Series B
July 30, 1982

Page No. ##

.END

ARBITRARY PAGE NUMBERS (.NUMBER DESIGN COMMAND)

When using the Print Command, T/Maker assumes that the number of the first page in any printing activity is one, the second page, two, etc. As we have already seen, the NUMBER option of the Print command can be used to establish a different number for the first page. For example,

WHAT NEXT? PRINT NUMBER 15 IT

prints the working area on the assumption that the first page is number 15, the second, 16, etc.

The ".number" design command could accomplish the same task by placing it as the first line in the file to be printed.

.NUMBER 15

However, the ".number" command can also be placed anywhere in a document to change page numbers in an arbitrary way. For example, the two design commands below would cause a new page (page number 9) to be started.

.NEW
.NUMBER 9

Unlike all other page numbers which are increased by one with each successive page, page number zero stays where it is until a new ".number" design command comes along. In long documents, it is often convenient to set the page number to zero immediately (i.e., a ".number 0" command at the head of the document) and let it stay there until the title page, table of contents, and other unnumbered pages are passed. With the start of the first numbered page, a ".number 1" command can be used to start normal sequencing. This might also bring into play top and bottom routines that lay dormant (see "Delaying the Use of Top and Bottom Routines").

To print a document that starts with 36 using page numbers at the bottom of pages beginning with page 37, the document might commence with the commands below.

.NUMBER 36
.BOTTOM 37

- # -

.END

FOOTNOTES

Footnotes are very easy to do.

Let us go about showing how to put footnotes into text by means of an example. Let us suppose that at the end of this paragraph, where the asterisk is, an author desired to write a footnote. If the following placement of the ".footnote" and ".end" commands were done, the footnote would appear at the bottom of the page, or, if there were insufficient room remaining on the page, at the bottom of the next page. Here is an exact version of the way it is done:*

.FOOTNOTE

* Having placed the ".footnote" command into the file, starting it in position one of a blank line, the author may now write the footnote in the zone between the ".footnote" and the ".end" commands. When in the editor, the footnote appears to be directly beneath the paragraph ending with the initial asterisk. But when the file is printed--either to the printer or to the screen--the footnote text will not appear below that paragraph at all, but at the bottom of the page, where a footnote ought.

.END

To repeat, directly above, are the footnote commands and footnote text as they would appear when viewing the footnote through the editor. But, down below, is the footnote as it will appear when the text is printed. The commands, of course, are not printed.

Incidentally, if one were to have both a footnote or two and a bottom routine, the bottom routine text would go to the bottom of the page, and the footnotes in the next available space higher up.

One line of space is added by T/Maker between the last line of regular text on a page and the first line of a footnote. No lines of space are added between footnotes, themselves, or between the last footnote and the start of the bottom routine. Of course, blank lines may be put within footnotes to achieve other spacing requirements.

* Having placed the ".footnote" command into the file, starting it in column one of a blank line, the author may now write the footnote in the zone between the ".footnote" and the ".end" commands. When in the editor, the footnote appears to be directly beneath the paragraph ending with the initial asterisk. But when the file is printed--either to the printer or to the screen--the footnote text will not appear below that paragraph at all, but at the bottom of the page, where a footnote ought.

BOLD AND UNDERSCORED TEXT

Hardware permitting (and most does), T/Maker provides for bold and underscored text.*

Bolding or underscoring text can be accomplished by means of either Toggle-Option or Design Commands. **

Toggle-Option Commands are the most appropriate for bolding or underscoring segments of text that cover only part of a line, whereas Design Commands are better suited for whole lines, particularly in tables.

THE TOGGLE-OPTION METHOD:

If T/Maker is in normal printing mode, touching the vertical bar key ("!") will initiate boldface text and touching the same key again will cease boldface.

If T/Maker is in normal printing mode, touching the back slash key ("\") will initiate underscored text and touching the same key again will cease underscoring.

FOR BOLD TEXT: "!" AT BEGINNING AND END OF BOLD SEGMENT

FOR UNDERSCORED TEXT: "\\" AT BEGINNING AND END OF SEGMENT

Of course, the "!" and "\\" signs, though they appear in the editor, will not appear in the file when printed, either to the printer or on the screen.

* The hardware and printer constraints on bold and underscored text are as follows: To use T/Maker's bolding and underscoring toggles-options, the printer must be able to backspace and overprint or be capable of returning the carriage without advancing to the next line. To use T/Maker's line-at-a-time bolding and underscoring commands the printer must be capable of returning the carriage without advancing to the next line. Some printers have self-contained capacity for bolding or underscoring--these often can be put into use by means of a "Printer Translation Table" placed into the T/Maker program file (see Tmodify Option 3 in the Customization Notes).

** Toggle-option commands--like the Insert Mode command (see Edit)--use the same command to turn something ON and to turn the same thing OFF.

There are a few strategies to keep in mind when placing the toggle characters in text. One strategy is to use them as separators between words, as in the case below.

<<

>>

Here the toggles are placed\between\words instead of spaces. This has the advantage that no extra space is required. However, this approach can create some\very long words\which can be a problem for the Align command in its attempt to justify text--especially, if the lines are short.

Another approach is to \use the toggles around the phrases\ to be bolded or underscored, but to still \leave a space\ between words. Align is \smart enough\ to hang the toggles over the end of the lines, since \they print as spaces.\ However, if the right margin is set in the first position, \Align\ may have no choice but to start \the line\with a toggle thus producing an unwanted blank space in the first position.

DESIGN COMMANDS:

T/Maker provides four Design commands useful for bolding and underscoring text.

UNDERSCORING COMMANDS:

- (that is, period followed by the underscore key) Underscores the previous line, beginning with the first nonblank character and terminating with the last nonblank character on that line.
- .= (that is, period followed by the equal sign) Underscore all nonblank characters in the previous line--in ordinary circumstance, this means only the words in a line are underscored.

BOLDING AND OVERPRINTING COMMANDS:

.+ **Bolds** the previous line.

.& **Overprints** the PREVIOUS line with the NEXT line.

That is to say, the text of line b, below,
will be printed on line a.

[line a] "Text text text text text text text."
. & [line b] "Text text text text text text text."

Both the underscoring and bolding commands follow the conventions of other Design Commands: they must begin in the first position with a period; and the line on which they appear does not print.

Incidentally, both the toggle-option and the design commands for bold and underscored text will generate repeated lines when printing to the screen because of the overprinting involved in both. This is normal.

ORGANIZING AND PRINTING LONG DOCUMENTS

CHAINING FILES CONTINUOUSLY:

Because the editor eventually runs out of space, long reports need to be divided into a number of files.

T/Maker provides an easy method for chaining or linking these files into what, for printing purposes, amounts to one long, continuous file. Thus linked, page-numbers, bottom routines, top routines and other Design Commands will apply and run continuously throughout the now-linked files, as if they formed one big file.

This linking is done by using the ".continue" command:

.CONTINUE "FILE NAME"

The command works like this: Suppose that we have just finished a report comprised of three chapters named A, B, and C--and each chapter is on its own file, of the same name.

Now, if we were to write the following on the last line of file A, the printer would go directly to file B and begin printing file B after file A had been completed:

.CONTINUE B

Likewise, the following command would start the printer on file C after B was completed, were the command placed on the last line of file B:

.CONTINUE C

INTERLEAVING FILES CONTINUOUSLY:

The ".continue" command will also interleave files--that is, permit one to print one or more files in the midst of another. Once more, the description is easiest with an example.

Suppose one had two files--File A and File B--and wanted to print File B after five paragraphs of File A, and then the rest of File A after File B's end.

This, as it turns out, is easily accomplished. The method is as follows:

1. Make File A the working file. Next, put a ".continue File B" command (starting in the first position) on the line in File A that File B is to begin.
2. T/Maker automatically goes back to the remainder of File A when File B has finished printing.

THE CONTINUE COMMAND'S GOVERNING PRINCIPLE:

It takes only a little imagination to see that chaining and interleaving capabilities of the ".continue" function could be orchestrated to produce a complex and Byzantine super-document, made up of segments drawn from all over the place on the disk.

In general, however, a long document is best handled by making one file which consists only of ".continue" commands and print design options common to the whole document (e.g., top or bottom routines). This file not only serves the purpose of joining pieces for printing, but with a few comments, it can be a handy table of contents to all the files that make up the document.

When and if one constructs documents in other ways or uses the continue command for other purposes, it is a good idea to keep "Continue's Principle" in mind:

CONTINUE'S PRINCIPLE: The working file can have lots of ".continue" commands sprinkled through it, but a file on disk can have only one ".continue" command, and that one can only appear as its bottom line.

THE PHYSICAL PAGE

This subsection is about commands that affect the physical dimensions of the page T/Maker prints.

Ordinarily, the user will not have to bother about these sorts of commands, since the physical dimensions of T/Maker's page are established and maintained by T/Modify (see T/Modify Option 4 in the Customization Notes) and apply to all printing sessions. In other words, T/Maker's assumptions about page-length and width are part of its store of fixed understandings, and not topics that are considered anew each time a new printing session comes along.

Nevertheless, T/Maker allows the user to override these same assumptions when a particular printing task requires new dimensions for the physical page. These override-commands apply only to the document in which they have been inscribed--more precisely, only to the lines that fall below the point the command appears.

Like other Design Commands, they begin with a period and are always started in the first position of an otherwise blank line.

PAGE SIZE

This command tells the printer the length of the page, in lines, from the top to the bottom of the paper. This, then, is not a measure of how many lines of print should go on a page, but rather a measure of how long the paper page itself is.

.PAGESIZE 66

The above command would tell the printer that the paper in use is 66 lines long.

Many printers also provide a self-contained ability to set the length of a page.

This command is ideal for preparing labels--which can be thought of as very small pages for this purpose. If a label sticker happens to be 7 lines long, pagesize would be set to 7. Perhaps the most economical method for preparing labels is simply to type the first label's contents, and, at the next line, give a ".newpage" command, sending the printer to the top of a new label, and then type the next label. Of course, the ".pagesize 7" command need only be given once, at the top of the file.

MODEL LABEL FILE:

```
.pagesize 7
Mr. Emile Nitrite
Chemical Bank
Falls Church, VA 22042
.new
Miss Rhoda Note
English Department
University of California, Berkeley
Berkeley, CA 94720
.new
Mr. Steve Adoor
Pier 35
The Embarcadero
San Francisco, CA 94101
.new

etc. . .
```

PAGE LENGTH

This command sets the number of lines T/Maker should put on a page. This command tells T/Maker to stop putting lines on a page after the nth line.

.LENGTH 60

It is the page-length value--whether the default value found in the T/Modify file or the value set with this command--that provides for T/Maker's pagination of documents. T/Maker travels through a file with its pagination "cookie-cutter" set at n-lines. Pages are thus "stamped" out for the printer.

At times one may find that a page's appearance might be improved by, say, finishing the last line of a paragraph on it, rather than taking the last line to the next page. This can be done by temporarily overriding the page length default setting with a ".length" command for that page. (Remember to reset the page length with another ".length" command on the next page.)

PAGE WIDTH

This command informs T/Maker of the width of the paper to be printed upon.

.WIDTH 80

Ordinarily, the document you are preparing will have margins less wide than the width of the paper to be printed upon. If, by chance, the user should set the margins wider than the page width, T/Maker will automatically "Panel" the printing task.

This means T/Maker will divide the printing session into two (or more) parts: First, the printer will print the portion of text falling between position 1 and position n , where n is the legal page width. Next the printer will start the printing over again from the top, this time printing the portion of text falling between position $n+1$ and position $2n$. The process will be repeated until the printing assignment has been completed. This feature of T/Maker is, of course, especially valuable in the construction of wide tables. Such panelling also permits the user to print documents wider than the printer's ordinary span of printing.

Incidentally, printed text that appears to have been cut short at the right-side margin is usually the result of a page-width setting that is lower than the right-side margin setting being used. In cases of panelling, T/Maker's Printer's Prompt will request permission to print each panel sector before the printing is started.

OTHER DESIGN COMMANDS

INDENTING BLOCKS OF TEXT (.INDENT)

This ".indent" design command shifts text, en masse, the designated number of spaces to the right on the page. It is mostly of use for shifting or centering the final production of tables and charts.

.INDENT 15

The above would shift text 15 spaces to the right. To cancel:

.INDENT 0

COMMENT LINES

Occasionally the user may desire to place comments into the file. The following command, beginning with two periods, is provided for that purpose:

.. THIS SHOULD BE TYPED AND IN THE MAIL BY WEDNESDAY.

Another use of this command is to suppress the printing of rows used in a table to hold intermediate results. Typically, the periods are begun in position 8 so that after the table is cleaned, the rows begun in this manner will be considered as comments and not printed.

NON-LINES

Putting a lone period in the first position of an otherwise blank line will cancel printing of the line. This command provides a handy barrier to the Align command (like all the other Design Commands), so that, for example, a "." command can be used to provide the demarcation between paragraphs without requiring the user to put a blank line between paragraphs. (Of course, a comment line could do the same thing, but it requires typing an extra period!)

NON-ALIGNING LINE

Putting a period in the first position of a line and following it with a comma makes a NON-Aligning Line. Unlike all other Design Commands, this line will print not showing the period and comma. It will not be aligned by the Align command.

.. The period and comma on this line would not be printed.

NON-aligning lines are useful as an alternative to putting in a special set of aligning wedges to stop the alignment of just a few lines. Of course, when this type of line is used, it is impossible to print characters in the first two positions of the line.

"CLEANING" TABLES

The design command ".clean" removes equations from tables prior to printing.

.CLEAN

To cancel the command later on in the file:

.NOCLEAN

This command provides a convenient alternative to using the Clean command as a response to the Universal Prompt. Unlike the Clean command which actually alters the file, the ".clean" design command produces the same result on the printer or screen without changing the contents of the file.

Other design commands following the ".clean" command may commence in position 1 or position 8.

PRINTING ALIGNING WEDGES

In order to force aligning wedges (see "Align") to appear in the printed text, use:

.WEDGES

To cancel the printing of aligning wedges:

.NOWEDGES

A MULTI-PAGE LETTER

top routine used from page 2 to provide headings, page numbers, and spacing to the first line of text	----> .top 2 John Doe ----- June 8, 1982	Page #
advance to line "x" to skip past stationary heading of first page	-----> .end -----> .7 ----> John Doe Doe Associates unaligned text and spacing as desired	June 8, 1982
	----- 1234 Main Street Centerville, Midstate 91106	
	-----> Dear Mr. Doe,	
aligning wedges for the body of the letter	-----> {{ All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy.	}} All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy.
text typed in "Drafting" Carriage Return Mode (the text shown is not yet justified--the Align command needs to be used to do this)	----->)){{ All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy.	
stop aligning for sign-off, etc.	-----> .block Sincerely, -----> Jack Dullboy 345 Hollow Road San Jose, California -----> .end	
sign-off made a "block" to ensure it is not split across two pages		

A SIMPLE APPROACH TO FORM LETTERS

```
    --> John Doe
1st name, address, and greeting    ----| Doe Associates
| 1234 Main Street
| Centerville, Midstate 10101
|
| --> Dear Mr. Doe,      %%
continue to the body of the letter  ----| .continue notice.7a
advance to a new page   ----| .new
|
| --> Bob Alpert
next name, address, and greeting  ----| Alpert & Son
| 451 University Drive
| Collegetown, NY 10516
|
| --> Alpy Boy!      %%
continue to the body of the letter  ----| .continue notice.7a
advance to a new page   ----| .new
|
| --> Queen Elizabeth
| Buckingham Palace
| London, England
|
| --> Your Royal Highness,  %%
| .continue notice.7a
| .new
```

Because this file includes multiple ".continue" commands, it must be the working file when printed.

The ".new" design command could be eliminated from the file and put as the last line in the letter.

Notice the double percent signs on some lines. These could be replaced with spaces using the Replace command before printing the form letters. They are used to flag the line that holds the greeting. Why? Well, to convert this file into a file suitable for printing labels or envelopes, one could use the command:

WHAT NEXT? DROP %% DROP .CONTINUE

A LONG DOCUMENT (MASTER FILE)

comment -----)	.. document A16, last revised June 29, 1981
special conditions of the physical -----)	.length 58
page (if any) -----)	.bottom 2
bottom routine used to print page -----)	- 8 -
numbers starting with page 2 -----)	
-----)	.end
page number set to zero for now* -----)	.number 0
continue with title page -----)	.continue title.a16
advance to a new page -----)	.new
continue with table of contents -----)	.continue contents.a16
advance to a new page -----)	.new
page number now set to one -----)	.number 1
continue with text of document -----)	.continue chap1.a16
comment -----)	.. blank page needed for map here
new page for "map" -----)	.new
one blank line printed -----)	
Chapter 2 to start on a new page -----)	.new
continue with next text file -----)	.continue chap2.a16
	.new
	.continue chap3.a16
	.continue chap3a.a16

Of course, this is just one of many of ways to organize a long document. Some of the design commands used here such as ".new" might have been placed in the "continued" files themselves.

* Remember that page number zero stays at zero. The page number must therefore be reset to another number to begin counting pages consecutively.

A LONG DOCUMENT (A TEXT FILE)

```
common aligning wedges placed at      ----)  {{(a))  
the top of the file and "tagged"  -----|  {{(c))  
with names      ----)  ((b))  
spacing as desired  -----)  
  
appropriate wedges "recalled"  -----)  ((=a  
                                         All work and no play makes Jack a dull boy. All work and no  
                                         play makes Jack a dull boy. All work and no play makes Jack a dull  
                                         boy. All work and no play makes Jack a dull boy. All work and no  
                                         play makes Jack a dull boy. All work and no play makes Jack a dull  
                                         boy. All work and no play makes Jack a dull boy.  
  
block defined here to ensure      ----) .block  
heading and first paragraph      -----| WORK VERSUS PLAY*  
will appear on same page  
                                         |  
                                         |  All work and no play makes Jack a dull boy. All work and no  
                                         |  play makes Jack a dull boy. All work and no play makes Jack a dull  
                                         |  boy. All work and no play makes Jack a dull boy. All work and no  
                                         |  play makes Jack a dull boy. All work and no play makes Jack a dull  
                                         |  boy. All work and no play makes Jack a dull boy. All work and no  
                                         |  play makes Jack a dull boy.  
-----> .end  
footnote placed as close as      ----) .footnote  
possible below the reference      -----| * see PLAY VERSUS WORK.  
to it      -----)  
  
aligning wedges recalled as needed  -----)  {{(c))  
                                         All work and no play  
                                         makes Jack  
                                         a dull boy  
  
                                         ((=a  
                                         All work and no play makes Jack a dull boy. All work and no  
                                         play makes Jack a dull boy. All work and no play makes Jack a dull  
                                         boy.
```

MIXING TEXT AND TABLES

the file will be cleaned as printed
(positions 1 through 7 will be
used only for instructions
related to tables)

-----) .clean

aligning wedges designed so that
no text will appear before
position 8 (i.e., the "cleaned"
area)

-----) ((())

All work and no play makes Jack a dull boy. All work and no
play makes Jack a dull boy. All work and no play makes Jack
a dull boy. All work and no play makes Jack a dull boy. All
work and no play makes Jack a dull boy. All work and no play
makes Jack a dull boy. All work and no play makes Jack a
dull boy. All work and no play makes Jack a dull boy.

)) ((

aligning stopped for the table -----)

indenting used to center the table -----)
-----)

.indent 10

.block

		WORK	PLAY	TOTAL
	ex	99.9	99.9	999.9
	ucl	+ +	+ +	= =
	+	Monday	12.0	5.0
	+	Tuesday	14.0	6.0
	+	Wednesday	21.0	2.0
				23.0
	avr	Average	15.7	4.3
				20.0

-----)
following text not indented -----)

.end

.indent 0

aligning starts again -----)

(())

All work and no play makes Jack a dull boy. All work and no
play makes Jack a dull boy. All work and no play makes Jack
a dull boy. All work and no play makes Jack a dull boy. All
work and no play makes Jack a dull boy. All work and no play
makes Jack a dull boy. All work and no play makes Jack a
dull boy. All work and no play makes Jack a dull boy.

T/MAKER: SPELLING CHECKER

TABLE OF CONTENTS

INTRODUCTION	IMMEDIATELY FOLLOWING
INITIALIZATION/INSTALLATION.....	SC-1
REFERENCE MATERIAL:	
A QUICK OVERVIEW.....	SC-2
SOME CONCEPTS.....	SC-3
PROOF COMMAND.....	SC-5
CORRECT COMMAND.....	SC-11
SPELL COMMAND.....	SC-17
AMEND COMMAND.....	SC-21

Introduction

Welcome to T/Maker's Spelling Checker!

What is a spelling checker program? As the name suggests, it's a program that checks for and lets you correct misspellings, typos, and uncommon words in a text file.

T/Maker's Spelling Checker program (like other spelling programs, incidentally) works by checking the words in a text file against all the words in a dictionary located on disk. Words found in the text file that were not found in the dictionary are then displayed for possible correction. All words found in this way are called "Unfamiliar Words" or "Unfamiliars" for short. T/Maker provides a number of other handy utilities as well!

T/Maker's Spelling Checker, for example, makes it easy to amend your dictionary, or work with more than one dictionary at a time (a good option by the way, for reasons we will see in a moment), or hunt out correct spellings with a dictionary's help.

The Checker is delivered from the factory with two dictionaries--the Standard Dictionary and the Extra Dictionary--together totaling roughly 52,000 words.

The Checker is comprised of four simple commands:

PROOF checks the words in your file against the words in one or more dictionaries and prepares a list of "Unfamiliar Words."

CORRECT lets you correct misspelled words.

AMEND lets you add or delete words from a dictionary.

SPELL helps you look up correct spellings in a dictionary.

This manual has two parts:

First, is an "Initialization Section" showing how to put the Checker into operation.

Second, is a "Reference Section" providing detailed descriptions of Spelling Checker commands.

Initialization

The T/Maker Spelling Checker commands are made up of the following files.

<u>Commands</u>	<u>Dictionaries</u>
PROOF.TMK	TMAKER.DIC
CORRECT.TMK	EXTRA.DIC
AMEND.TMK	
SPELL.TMK	

These four command files consume approximately 40K bytes of disk space. TMAKER.DIC consumes about 26K and EXTRA.DIC consumes about 78K bytes of disk space.

The Spelling Checker provides two, rather than one, dictionaries. Two are provided primarily for the benefit of users with limited disk space. You may decide to use only TMAKER.DIC as your dictionary--in that case, only that file need be copied to your working disk. TMAKER.DIC is a complete and usable basic dictionary.

If you have plenty of disk space, we suggest you copy only the commands (and not the dictionaries) to your T/Maker working disk. Next, with T/Maker running on drive A and the Spelling Checker disk in drive B, give the following command:

1/1 WHAT NEXT? AMEND FROM B:TMAKER + B:EXTRA TO A:TMAKER

This command will merge the two dictionaries on drive B into a single new dictionary on drive A. This new dictionary will have the name TMAKER.DIC. The resulting dictionary consumes approximately 100K bytes.

One may also elect to work with the two dictionaries separately. You may use the command "PROOF" to check your text against the TMAKER.DIC dictionary and the command "PROOF WITH EXTRA" to check text against both dictionaries. This approach may prove to be more efficient than working with a single big dictionary in some circumstances.

A Quick Overview

The Spelling Checker is for checking text for misspelled words, typos, and uncommon words.

It works by checking all the words in a body of text against the contents of a dictionary.

You use the PROOF command to check text. The command works on the Working Area only (that is, the area of the Working File below the cursor when the Editor was quit).

PROOF reports a Correction List of all words in the text not found in the dictionary. This List is temporarily inserted at the top of your text. It looks like this:

```
!= rrose      rrose
!= hte       hte
!= homologous homologous
```

The Correction List shows two samples of these words side by side. If a word is misspelled:

First, you change the second sample of the word (with the EDITOR) to the correct spelling.

Second, you return the cursor to the top of the Correction List and invoke the CORRECT command. T/Maker will now correct the spelling of that word wherever, and in whatever capitalization it may appear, everywhere in the text.

If a word is not misspelled (that is, it was placed on the Correction List simply because it did not exist in the dictionary), leave the word's second sample unchanged. The CORRECT command will leave this word unchanged in the text as well.

You may on occasion wish to add such words to your dictionary. This is done by using the AMEND command (see instructions herein).

Now and then you may also be unsure of a word's correct spelling. In such cases the SPELL command may be of some help (see instructions herein).

Some Concepts

Before describing the Checker's commands, we need briefly to consider two basic concepts: "word" and "dictionary".

WHAT'S A WORD?

The Spelling Checker's PROOF command searches your text file looking for words that do not appear in T/Maker's dictionary. To do this, of course, T/Maker must have an operational conception of what a word is. The decision is a fundamental one, and it is a good idea for the user to know how T/Maker goes about making it.

T/Maker defines a "word" as a series of one or more contiguous alphabetic characters. The characters may be in upper case, lower case, or mixed cases. T/Maker ignores case when checking text file words against a dictionary--thus, for example, "ThErE" is the same word as "there" for look-up purposes. (Consequently, T/Maker's spelling commands cannot be used to check capitalization.)

A word may contain an apostrophe within it. So, for example, "hasn't" or "wouldn't" are considered words. Apostrophes appearing at the beginnings or the ends of words are not considered part of a word--for example, "users'" is seen simply as "users".

Apostrophes followed immediately by an "s" are also ignored. T/Maker sees "man's" simply as "man". An apostrophe followed by an "s" is regarded as a possessive form and ignored for look-up purposes.

Any nonalphabetic character that is not an apostrophe is seen as separating words. So, for example, "easy-to-use" is seen as three separate words and "3rd" is seen as the word "rd".

Because such rules of definition will never fully reflect the complexity of actual language, you may find a few "surprise words" when you use the Spelling Checker commands. You shouldn't have difficulty spotting them, though. If necessary, you can simply use the Checker's AMEND command to add such words as "rd" or "st" or "Mr" to your dictionary.

WHAT'S A DICTIONARY?

The dictionary is a collection of words.

T/Maker's Spelling Checker delivers two dictionaries to you: the STANDARD dictionary (file name, "TMAKER.DIC") and the EXTRA dictionary (file name, "EXTRA.DIC"). Unless you specify

otherwise, T/Maker's Checker commands automatically consult the STANDARD dictionary only. You call up the EXTRA dictionary or additional dictionaries you may have created yourself by using a command parameter.

The STANDARD dictionary contains about 15,000 words; the EXTRA, about 35,000 additional words.

A bigger dictionary, however, is not always a better dictionary. A bigger dictionary takes more space on disk and slows down the PROOFing process. Also, more entries in a dictionary mean that more words pass unchallenged through the PROOFing process. For example, although "ai" is indeed the word for a three-toed sloth, you may in fact be considerably more likely accidentally to have left off the "m" in "aim" than actually to have intended to write "ai"!

The best dictionary for you is the one that consists of all the words you actually use. Needless to say, such a dictionary cannot be provided readymade by the manufacturer.

You may wish to build your own dictionary from scratch. This can be FUN. By using the Spelling Checker commands on text you have written in the past you may be able to build a personalized dictionary quite quickly.

The PROOF Command

DESCRIPTION

The PROOF command checks the words in the working file against the words in one or more dictionaries.

1/1 WHAT NEXT? PROOF

Only the Working Area (see T/Maker III's Reference Manual for the definition of "Working Area") is searched or "proofed".

The PROOF command generates a three-part report called "PROOF'S REPORT". Some elements of this report are temporarily inserted into the text file. These elements are:

- (1) A list of all text-words that were not found in a dictionary. These words, you'll recall, are called "Unfamiliar Words" or "Unfamiliars".
- (2) A special flag (described below) is placed at every occurrence of an Unfamiliar Word in the text itself.
- (3) A variety of statistics concerning the PROOFED text file are reported (described below).

As we will see, the CORRECT command serves to remove these various reports from the working file (the editor may be used to remove the reports as well--see below).

When you use the PROOF command-word by itself, T/Maker checks only the STANDARD (TMAKER.DIC) dictionary. You may, however, follow "PROOF" with a parameter word instructing T/Maker to use other dictionaries as well. This is done by including the names of desired additional dictionaries as shown below.

1/1 WHAT NEXT? PROOF WITH EXTRA WITH ACRONYMS

Notice that each additional dictionary is preceded by the word "WITH", which tells T/Maker that a dictionary name follows. When using PROOF with multiple dictionaries one should specify all the dictionaries you want searched in a single command.

Incidentally, all dictionary files should have names with the suffix ".DIC". When you create a dictionary of your own it should be given a name with that suffix: such as file names, "ACRONYMS.DIC" or "FLOWERS.DIC". When asking for a search of a dictionary, however, it is not necessary to mention the ".DIC"

name-suffix in the command parameter. Thus, in the example above, T/Maker would automatically look for dictionaries named "EXTRA.DIC" and "ACRONYMS.DIC". The PROOF command always searches its STANDARD dictionary first (as long as TMAKER.DIC is available on disk), even when additional dictionaries are requested.

Though your dictionaries may normally reside on the same disk drive as the Spelling Checker commands, it is possible to place additional dictionaries on other disk drives. In such instances, the dictionary name should be preceded by a drive specification. For example:

1/1 WHAT NEXT? PROOF WITH B:ACRONYMS

WORD LISTS

A "Word List" is a small dictionary used when you have some special proofing requirements.

Suppose, for example, that that you were writing a report that contained a number of persons' names and departmental names, too. You might not want to add these names to your permanent dictionary.

The best solution is to construct a "Word List" made up of the names in question.

How is such a "Word List" constructed?

- (1) Begin by CREATING a "Word List" file called, say, "NAMES.DIC" or "DEPTS.DIC" (just as dictionary file names, "Word List" file names should have ".DIC" suffixes).
- (2) Key in a vertical list of special words--in this case, the words making up the names of persons and departments to be mentioned in the report. A number of points need to be made about this list:
 - (a) For purposes of the Word List, each multi-word name becomes a series of single-word entries. So, for example, the name "June Marie Richards" would be entered as the three separate words "June", "Marie", and "Richards". (Do not use quotation marks in the entering of Word-List words.)
 - (b) All words on the list appear on their own line, and all words begin in column-position 1.
 - (c) All words must be entered in all-caps or all-smalls, even the names of persons or other proper nouns that would normally include initial

capitals. This requirement will not adversely affect searches of the Word List.

- (d) The Word List must be alphabetized from A ascending. (You can use the SORT command, of course, to carry out such alphabetization once the list's elements have been supplied--see the SORT command in the T/Maker III Reference Manual.)
- (3) Don't forget to SAVE the Word List file once it has been CREATED, supplied with entries, and SORTED!

A Word List named NAMES.DIC would be searched (along with the STANDARD dictionary) by the following command:

1/1 WHAT NEXT? PROOF WITH NAMES

THE PROOF COMMAND'S THREE REPORTS

REPORT NUMBER 1: A LIST OF UNIDENTIFIED WORDS

PROOF adds a complete list of Unfamiliar Words to the top of your working area.

This list has a special form. Each Unfamiliar Word is presented on its own line. Each line is called a "Correction Line".

Each Correction Line has three elements:

- (1) It begins with an equal sign followed by an exclamation point, these in column-positions 1 and 2.
- (2) & (3) Next, come two samples of the Unfamiliar Word, side by side.

A typical Correction Line might look like this:

```
=! agan      agan
```

You will have as many correction lines inserted at the top of your working area as you have Unfamiliar Words in your file.

The first sample of the Unfamiliar Word on the correction line provides the "current spelling of the word".

The second sample of the Unfamiliar Word provides "the word sample you will (if correction is necessary) correct".

When PROOF makes its report these two word samples are identical. YOU CORRECT THE SECOND SAMPLE (with the T/Maker Editor) AND THEN INVOKE THE "CORRECT" COMMAND TO CORRECT THE TEXT.

If an Unfamiliar Word in the Correction List happens to be correctly spelled--that is to say, the word appears in the list simply because it did not appear in the dictionary--then LEAVING THE SECOND SAMPLE OF THE WORD AS IS AND INVOKING THE CORRECT COMMAND WILL SERVE TO LEAVE THE WORD AS IS IN THE TEXT AS WELL.

REPORT NUMBER 2: THE FLAGGING OF UNIDENTIFIEDS IN THE TEXT

The PROOF Command flags all unfamiliar words in the working area by preceding them with a question mark and an equal sign--that is, ?=.

For example, the word "agan" in the first sample (below) would appear as it does in the second sample after the PROOF command had done its work:

1st Sample: Here I go agan.

2nd Sample: Here I go ?=agan.

This flag is provided in order that you can more readily find occurrences of Unfamiliar Words in the text. The flag helps locate such words simply by making them more noticeable to a visual scan and also by making a "Search for a String" keystroke command (see T/Maker Editor description in the Reference Manual). If you want to see the various occurrences of a word in context, you simply search for "?=agan", which approach will avoid the discovery of such words as "Reagan" or "pagan" in your search.

Invoking the CORRECT command automatically clears the text of PROOF'S flags.

REPORT NUMBER 3: THE STATISTICAL REPORT

The PROOF command presents a brief statistical report about your text at the upper right of your screen. This report looks something like this:

1149	TOTAL WORDS
479	UNIQUE WORDS
35	UNFAMILIAR WORDS
17	WORDS/SENTENCE AVERAGE
4.7	CHARACTERS/WORD AVERAGE

TOTAL WORDS gives the count of all words in the working area.

UNIQUE WORDS gives the count of all words that are unique character strings in the working area.

UNFAMILIAR WORDS gives the count of all words in the text that were not found in the dictionary or dictionaries that were searched.

WORDS/SENTENCE AVERAGE gives the average number of words per sentence in the text that has been PROOFED. This, it should be noted, is only a rough estimate. T/Maker calculates mean sentence length by dividing the number of words by the sum of the number of periods, question marks, and exclamation points in the text. Thus, headings, table titles, and the like will skew the reported mean.

CHARACTERS/WORD AVERAGE gives the mean number of alphabetic characters per word.

The statistical report disappears from the screen when the CORRECT or EDIT command is used.

CAPITALIZATION

As we've seen, the PROOF command scans text for unfamiliar words. Naturally, it may find a single unfamiliar word in a variety of forms of capitalization. For example, suppose the following sentence appeared in the text:

Agan and agan, I screamed "AGAN! aGan!"

T/Maker regards all these instances of "agan" as the same word, and it will present the word in only one form of capitalization in the Correction List. But which capitalization does it present?

T/Maker follows these rules of preference:

1st Choice: An instance of nonstandard capitalization--in this case, the "aGan". This is done so that nonstandard capitalizations will more surely be brought to the user's attention.

2nd Choice: All smalls--"agan".

3rd Choice: Capitalized first letter, all smalls thereafter--"Agan".

4th Choice: All caps--"AGAN".

(See the description of the CORRECT command for further information on capitalization.)

BUILDING A VOCABULARY LIST

Suppose that you had occasion to want to build a vocabulary list based upon a particular text.

As it happens, T/Maker's PROOF command provides an easy way to do this.

Recall that PROOF goes about its job by checking every word in a body of text against a dictionary. All words not in the dictionary are then presented in the Correction List.

Well, it follows that if the dictionary contains no words, then all unique words appearing in the text will be presented in the Correction List. This, in other words, is equivalent to a complete vocabulary list for a given text.

You can accomplish this end most easily by running the PROOF command on a disk that contains no TMAKER.DIC file. PROOF, finding no dictionary at all, will present all words as unfamiliar.

This technique may produce some problems for files with many unique words. The Correction List may exceed available memory, or the PROOF command may not be able to keep track of so many unfamiliar words. In such cases it is necessary that you split the working file into smaller files.

MESSAGES

PROOF generates a number of self-explanatory error messages.

The CORRECT Command

DESCRIPTION

The PROOF command generated a list of Unfamiliar Words. The CORRECT command provides the means for either correcting misspelled words or restoring already correctly spelled words unchanged to the text.

1/1 WHAT NEXT? CORRECT

As we have already seen, the PROOF command generates a list of Unfamiliar Words at the top of the Working Area.

This list will now provide the medium for correcting misspelled words. One makes spelling corrections by altering the spelling of the second sample of the unfamiliar word in the Correction List. One makes this alteration simply by using the T/Maker EDITOR. Once all alterations have been made one should return to the top of the Correction List, quit the EDITOR, and invoke the CORRECT command. T/Maker now corrects spellings, removes flags from the text.

As it happens there are three circumstances the user may face when he sees an Unfamiliar Word on the Correction List: (1) the word is obviously misspelled and requires correction; (2) on the other hand, the word might be misspelled, but the user needs to check a dictionary to make sure of it and to find out the correct spelling; or, finally, (3) the word is correctly spelled and was put on the Correction List simply because it did not appear in the dictionary or dictionaries that were searched (in this 3rd case, you may wish to add the unfamiliar word to the dictionary--see the AMEND command).

Let's consider these three circumstances in turn.

THE WORD IS OBVIOUSLY MISSPELLED

Here is an example of an obvious misspelling:

```
=! approximatley approximatley
```

Here, the "l" and "e" have been inverted.

When words are obviously misspelled, use the EDITOR to correct the second sample of the Unfamiliar Word. Don't change the first sample. You may use any of T/Maker's editing keystrokes to effect this change. (A REPLACE command, however, will have the obvious disadvantage of replacing both the second and the first

instance of the sample word and so should not be used.)

=! approximatley approximately

Using the CORRECT command will now substitute the second sample's spelling for the first sample's spelling in the text. It will also remove the ?=tags from word-occurrences in the text.

THE SITUATION IS UNCLEAR

Sometimes you will not know whether an Unfamiliar Word is actually correct or not. You may encounter two kinds of ambiguity: (1) the unfamiliar word may appear incorrect but in fact be correct in its proper context--for example, it may be the "rd" in "3rd"; or, (2) the word may be misspelled but your knowledge of spelling is such that you are not absolutely certain of it. In instances of the first kind, you will want to examine the word in context. In instances of the second kind, you may want assistance from the dictionary in establishing the word's correct spelling.

(1) Instances of the First Kind--Suppose that the following characters appeared on your Correction List:

=! tht tht

You're not sure of the status of this string. You now have a number of choices.

First, you can use the "Search for a String" keystrokes to locate occurrences of the "?=tht" string in the text. You may simply correct offending instances as you encounter them in the text (remember, however, this procedure would bypass an occurrence of "?=THT"). A better approach would be to return to the string in the Correction Line and alter the text en masse.

Secondly, you may have occasion to desire to ignore ambiguous strings until you have corrected or restored all the unambiguous words in the Correction List. You may thus postpone consideration of the ambiguous word by simply erasing the second sample of the word on the Correction Line, like this:

=! tht

Using the CORRECT command now will correct other words in the Correction List but completely ignore occurrences

of "tht". The "tht" Correction Line will remain at the working area's top. Text occurrences of "tht" will not be removed and nor will their "?=" tags be deleted. In this way you employ the CORRECT to correct unambiguous words while keeping ambiguous words on hold for a later effort.

(2) Instances of the Second Kind--Suppose that the following character-string appeared on your Correction List:

```
=! indigestable indigestable
```

You have a hunch--but only a hunch!--the word is misspelled. T/Maker can now help you find out the word's correct spelling. You may use the "List Mode" of the SPELL command (see SPELL command).

THE WORD IS CORRECT

Many correctly spelled words simply won't be in your dictionary. For example, consider the correction line below:

```
=! adjutancy adjutancy
```

This word is correctly spelled, and let's assume that you actually meant to use it. It was not in your dictionary, and so the PROOF command has placed it into the Correction List. You now will want to leave the word unchanged in your text, and you may want to add this word to one or another of your dictionaries.

In either case you must use the CORRECT command to remove this word's ?=tags in the text. You can leave the Correction Line as is (or delete the Correction Line) in order to remove the ?=tag with the CORRECT command.

When the CORRECT command is thereafter used the ?=tags and the word's Correction Line will be removed.

If follows from this that using a CORRECT command after a PROOF command without making any changes in the second samples of Correction Lines will always restore a file to its original, pre-PROOFED form.

Adding a word to your dictionary is accomplished in two steps. First, replace the Correction Line's exclamation point with a plus sign, like this:

```
+! adjutancy adjutancy
```

(If CORRECT is now used, it will remove the ?=tags on "adjutancy" in the text, but the command will ignore "adjutancy's" Correction Line. This means you can safely use the CORRECT command before you turn to adding words to a dictionary.)

Second, you invoke the AMEND command.

CAPITALIZATION

We saw earlier that the PROOF command needed to decide which capitalization-form of an unfamiliar word it would present in the Correction List. The CORRECT command has a different responsibility: It must decide how to handle the capitalization of words whose spelling has been corrected.

CORRECT'S general rule is to follow the capitalization that appears in the text. For example, if we corrected "agan" to "again" in the following sentence,

Agan and agan, I screamed "AGAN!"

it would emerge from the CORRECT command's attentions as,

Again and again, I screamed "AGAIN!"

For the three standard uses of capitalization (all caps, all smalls, and first letter only capped), CORRECT mimics the capitalization of the text word-instance it finds.

When a word has unusual capitalization, however, a new rule takes over. In this case, the CORRECT command will use the exact capitalization you provided on the Correction Line. Recall that T/Maker tries to make you aware of odd instances of capitalization by putting them (when they exist) on the Correction Line.

Consider the following example. PROOF has placed the word TMakr on the Correction List:

```
=! TMakr TMakr
```

If you correct this word to "TMaker" in the Correction List, as follows,

```
=! TMakr TMaker
```

then the CORRECT command will make the following substitutions:

"tmakr" will become "tmaker" (the all-smalls case)
"Tmakr" will become "Tmaker" (the first-letter-only-cap case)
"TMAKR" will become "TMAKER" (the all-caps case)

Any and all other capitalization-forms of "TMakr" will become "TMaker".

You may on occasion want to substitute only one form of capitalization for all existing forms currently in the text. For the previous example, suppose you wanted "Tmaker" and nothing else regardless of how the word is currently capitalized. You can accomplish this task by surrounding the second sample of the word on the Correction Line with quotes (double or single), like this:

```
=! TMakr "Tmaker"
```

These quotes tell the CORRECT command: Use the replacement spelling exactly as I have given it for any occurrence of the original word. Thus correct would substitute:

```
"Tmaker" for "tmakr"  
"Tmaker" for "Tmakr"  
"Tmaker" for "TMAKR"  
"Tmaker" for any and all other cases.
```

You may also put quotes (single or double) around the first word-sample in the Correction Line. In this case, only the specified form of capitalization will be replaced and the replacement word will be capitalized exactly as it is in the second sample-word.

```
=! "TMakr" Tmaker
```

The above line would cause "Tmaker" to be substituted for "TMakr." No other substitutions would result from this line; "TMAKR" would not be replaced by the new spelling or capitalization.

SUBSTITUTING A PHRASE

You may substitute a phrase for a word by simply putting the phrase in the second word place. Quotation marks are not required. below).

```
=! inbetween in between  
=! you party of the first part
```

OTHER USES

You may use CORRECT without first using PROOF. This is done by creating your own Correction Lines (following the conventions T/Maker itself follows in such Lines). When using a CORRECT command from scratch it is not necessary to put ?=tags ahead of words to be corrected. Correction Line entries, however, must begin with "=!". These characters are followed in turn by the current word and the corrected word or phrase.

The SPELL Command

The SPELL command looks up word-spellings in a dictionary. It is T/Maker's effort to solve the ancient dilemma of needing to know the spelling of a word before one can look it up in the dictionary!

The word (or words) whose spelling you want the SPELL command to check is called the "Ambiguous Word Reference".

The command lets you look up a word whose spelling you don't know by putting "wild-card" symbols in the places of the word's letters you don't know.

For example, if you didn't know whether "correspondence" is spelled with an "ence" or "ance" you could describe the word as "correspond?nce". T/Maker's SPELL command could then look it up for you.

The ? in correspond?nce is a wild-card symbol. T/Maker provides two wild-card symbols-- ? and * : The ? means that this symbol stands for a single letter at this place in the word; an * means that one or more letters may be sited here in the word--including the possibility of no letter at all.

You place ?'s or *'s at the spots of unknown letters in the "ambiguous word reference".

SPELL'S TWO MODES OF OPERATION: THE COMMAND MODE AND THE LIST MODE

The SPELL command can be operated in two different modes: the first is called the "Command Mode" and the second is called the "List Mode".

The Command Mode works from T/Maker's Universal Prompt. The List Mode works from a Correction List generated by a PROOF command (or generated by oneself). Let's consider these two modes in turn.

The Command Mode:

The Command Mode may be used simply by answering as below to the Universal Prompt:

```
1/1 WHAT NEXT? SPELL [ambiguous-word-reference]
```

When the SPELL command-word is used by itself the SPELL command looks up spellings in the STANDARD (or TMAKER.DIC)

dictionary only. You may, however, instruct the command to use a different dictionary by adding the word "WITH" followed by the name of the dictionary you want consulted, like this:

1/1 WHAT NEXT? SPELL [word-reference] WITH dictionary-name

A dictionary must have a file-name with a ".DIC" suffix. Although dictionaries ordinarily will reside on the same disk drive as the Spelling Checker commands, it is possible to use a dictionary on another drive. In such instances, the dictionary name must be preceded by a drive specification, like this:

1/1 WHAT NEXT? SPELL [word-reference] WITH B:ACRONYMS

The results of SPELL's search for words that fit your specifications are temporarily reported at the top of the screen.

SPECIFYING THE "AMBIGUOUS WORD REFERENCES"

You must provide the SPELL command with an "ambiguous word reference" describing the word-candidates to be sought in the dictionary.

As noted, a question mark signifies one unknown letter. For example, the ambiguous word reference cited below would send the SPELL command looking for all four-letter words that begin with "i" and end with "d".

i??d

An asterisk signifies any number of unknown letters, including no letter at all. For example, the ambiguous word reference cited below would send the SPELL command looking for all words starting with "r" and ending with "ing". The harvest of dictionary words would include "ring" as well because the asterisk may stand for no letter at all.

r*ing

Ambiguous Word References may be made up of multiple question marks and asterisks although quicker results depend on tighter specifications. Because dictionaries are alphabetically ordered, providing the first letter of an ambiguous word reference will dramatically reduce the searching time.

The SPELL command puts the words found on the screen. If more than one screen is needed, it will prompt you at the end of each screen as to whether you wish to continue or not. Note that the command "SPELL *" can be used to report the entire dictionary in this way.

THE LIST MODE

Used in its "List Mode", the SPELL command looks up possible spellings for words described in a Correction List. This Correction List may have been generated by the PROOF command or by one's own creation.

In this case, one places the wild-card symbols into the second occurrence of Correction Line words. The SPELL command is then followed by the word "IT", like this:

```
1/1 WHAT NEXT? SPELL IT
```

This usage may also include the word "WITH" followed by a dictionary name if a dictionary other than "TMAKER.DIC" is to be used.

When the "IT" option is used, the SPELL command first searches the working area for Correction Lines that contain ambiguous word references. When such a word is found, SPELL searches the dictionary for word-candidates. The harvest of candidates is put to the right of the original Correction Line word (until the line is full).

Let us consider once more the example we described in our discussion of the CORRECT command. Suppose that after using PROOF, you find yourself with the correction line below.

```
=! indigestable indigestable
```

You suspect the word is misspelled but you are not certain. Here, you might consider making the second Correction-Line word into an ambiguous word reference, as below:

```
=! indigestable ind*est*ble
```

Using the "SPELL IT" command would now place word candidates from the dictionary on the same Correction Line:

```
=! indigestable ind*est*ble indestructible indigestible
```

As can be seen, "indigestable" is actually spelled "indigestible". You may now put the correctly spelled word into the second place on the Correction Line (erasing the others) and invoke the CORRECT command.

Incidentally, one may make an ambiguous word reference of the first Correction-Line word--this will look up word candidates.

```
=! li*en*e
```

The "SPELL IT" command will now yield:

```
=! li*en*e licensable license licensee licentiate
```

The AMEND Command

DESCRIPTION

The AMEND Command adds or deletes words from a dictionary or combines dictionaries.

1/1 WHAT NEXT? AMEND

Whenever a dictionary is updated with the AMEND command, the dictionary's previous "un-updated" version is changed to a back-up file (named-suffixed ".BAK"), in the fashion of the SAVE command.

Dictionaries are very special files. They have been compressed to allow for many words. Thus, you cannot GET them or EDIT them as other files. You must use the AMEND command to change them.

When the AMEND command-word is used by itself, it updates the STANDARD dictionary (TMAKER.DIC) only. Other uses and updates of other dictionaries involve additional parameter words in the command.

UPDATING A DICTIONARY

As described earlier, when the PROOF command finds Unfamiliar Words in text it lists them on the Correction List. Each line in this Correction List begins with the characters "=!". The AMEND command begins with a Correction List.

There are two simple steps involved in adding or deleting a word from a dictionary:

First, one must change the "=!" symbol at the beginning of the Correction Line to either a "+!" (for "add this word") or "-!" (for "delete this word"), like this:

+! adjutancy adjutancy	-- word to be added
-! preceeding	-- word to be deleted

Second, one returns the cursor to the top of the Working Area, quits the Editor, and gives the AMEND command.

T/Maker will now take a few moments to update the dictionary.

Only the first word on a Correction Line is added or deleted from a dictionary.

OTHER FORMS OF AMEND

AMEND has a number of additional uses:

Creating a New Dictionary

AMEND can be used to create a new dictionary directly from a Correction List. To do this, give the following command when the desired Correction List is in the Working Area:

1/1 WHAT NEXT? AMEND TO [new dictionary-name]

If no disk drive specification is given in the new dictionary's name, AMEND assumes the dictionary should be lodged on the same drive the AMEND command is lodged.

T/Maker, incidentally, will automatically add a ".DIC" suffix to any name you give a new dictionary, unless you have imposed a suffix yourself.

Updating Other Dictionaries

AMEND allows you to update dictionaries other than the STANDARD one.

In order to update another dictionary simply give the following command:

1/1 WHAT NEXT? AMEND FROM [dictionary-name]

You may have occasion to desire to AMEND a dictionary and change the name of the newly-amended version. This, in other words, is a means for creating a new dictionary out of an old one.

To do this, simply give the following command:

1/1 WHAT NEXT? AMEND FROM [old dictionary-name] TO [new name]

Combining Two Dictionaries

You may combine two dictionaries by using the AMEND command as follows:

1/1 WHAT NEXT? AMEND FROM [dictionary-name 1] + [dictionary-name 2]
TO [new dictionary-name]

If the Working Area contains a Correction List, the combined dictionary you create will also include any additions or subtractions specified in that Correction List.

Subtracting One Dictionary From Another

AMEND permits you to subtract the words of one dictionary from another. The following command would subtract "dictionary 2's" words from "dictionary 1's" words:

1/1 WHAT NEXT? AMEND FROM [dictionary-name 1] - [dictionary-name 2]
TO [new dictionary-name]

This command generates a new dictionary containing the words in dictionary 1 not contained in dictionary 2.

If the Working Area contains a Correction List, the resulting dictionary will also include any additions or subtractions specified in that Correction List.

TIME CONSIDERATIONS

AMENDING a large dictionary is a time-consuming process.

We suggest a way to save some of this time.

Let's suppose that over the course of a day's work you will generate ten different Correction Lists, each with words that should be amended to your STANDARD dictionary.

Doing ten separate AMENDS, however, will be slow-going.

Therefore, we suggest that you begin a new day by creating a dictionary called TEMP.

Do this when you are working on the first Correction List of the day, with the following command:

1/1 WHAT NEXT? AMEND TO TEMP

This command, of course, will create a dictionary called "TEMP.DIC". TEMP.DIC will have the chief virtue of being a small dictionary. Therefore, it can be updated very rapidly. The idea is to use this small dictionary during the course of the day to

collect new words you want ultimately to AMEND to your larger dictionary.

In PROOFS of successive texts you will want to add the TEMP dictionary to the proofing process, like this:

1/1 WHAT NEXT? PROOF WITH TEMP

With the TEMP dictionary included in the proofing process in this way, words already in the TEMP dictionary will not be duplicated in successive Correction Lists.

Add new words to the TEMP dictionary by using the following command:

1/1 WHAT NEXT? AMEND FROM TEMP

Now, at the end of the day or when sufficient time is available, you may update your STANDARD dictionary with the TEMP dictionary's interim collection of new words. Just use the following command:

1/1 WHAT NEXT? AMEND FROM TEMP + TMAKER TO TMAKER

This command will add TEMP'S words to the STANDARD dictionary's (TMAKER.DIC) words, creating a new, updated STANDARD dictionary.

This method will save a good deal of time in most circumstances.

SPACE CONSIDERATIONS

Long dictionaries take up lots of disk space. If the two dictionaries delivered with the Spelling Checker package are combined, the result is a file of approximately 100,000 characters (not words!).

Here are a number of suggestions for keeping your disk from becoming overcrowded.

First, you may want to delete any old ".BAK" dictionaries lying around on your disk.

In some cases, it may be necessary to AMEND so that the resulting dictionary is placed on a different disk. For example,

assuming your commands and dictionaries were on Drive A, the following command would put the AMENDED dictionary on Drive B.

1/1 WHAT NEXT? AMEND FROM TEMP + TMAKER TO B:TMAKER

You might then wish to delete the un-updated version of the STANDARD (TMAKER.DIC) dictionary from Drive A. Use the copy command of your operating system to copy the new TMAKER.DIC from Drive B back to Drive A. You could also use the command below to make this move across disks--though it would be a good deal slower:

1/1 WHAT NEXT? AMEND FROM B:TMAKER TO A:TMAKER

T / MAKER : DATABASE

TABLE OF CONTENTS

INTRODUCTION	IMMEDIATELY FOLLOWING
REFERENCE MATERIAL:	
INITIALIZATION.....	DB-1
OVERVIEW OF STRUCTURE AND OPERATION.....	DB-2
RECORD/FORM DEFINITION.....	DB-3
UPDATE COMMAND.....	DB-19
SET COMMAND.....	DB-25
ORDER COMMAND.....	DB-37
GROUP COMMAND.....	DB-39
SELECT COMMAND.....	DB-43
COMPLETE.....	DB-51
LIMITS.....	DB-54
EXAMPLES OF DATABASE.....	DB-45

T/Maker's Data-Base System

Welcome to T/Maker's new data-base system.

We think it will provide you with an easy-to-use and yet powerful tool for tackling data-base problems. [1]

What is a data-base system?

It's really nothing more sophisticated than a set of 3X5 index cards in a card-file box.

In a conventional card-file box you put pieces of information on each card. You update the information on a card now and then, you throw away a card you haven't used in years, or you add a new card sometimes. Sometimes you need the information on a particular card or a set of cards. Sometimes you may want to make a report based on information drawn from all the cards.

A data-base file does the same sorts of things--and more. As we will see, a data-base file "in" your computer has a number of powerful advantages over a conventional card-file box.

How does T/Maker's data-base system work?

T/Maker's data-base system begins with an ordinary text-file that performs as a "card-file box." All your "cards," and all their information, will be placed into this file.

You transform an ordinary text file into a data-base file by showing T/Maker what a standard "card" of information should look like. In other words, you must show T/Maker the format of a "card" or (in computer lingo) a "record."

You do this by designing the record yourself, using the T/Maker Editor. The "card" or "record" format is created and stored in a text file. There are some rules for how this record should be laid out, which you'll need to learn.

[1] As with all T/Maker files, a data-base file must fit in the working memory of your computer. There are some techniques which can be used to tie data-base files together for particular purposes, but we do not recommend these commands for problems involving massive amounts of data (ie. on an IBM PC, you can put about 400 names and addresses on one file and chain files together to store about 1200 names and addresses. But we would not recommend trying to build a file with 10,000 names and addresses).

Once supplied, the record format makes the new T/Maker data-base commands available to you:

The UPDATE command lets you enter, revise, or delete data. It is the "Editor" of the data-base system and has many utilities as we'll see.

COMPLETE transfers data "in bulk" from one file to another.

ORDER puts records into alphabetical or numerical order.

GROUP makes summary reports from a data-base.

The SELECT command provides two utilities:

- (1) It lets you select subsets of cases from the data-base file.
- (2) It lets you reformat data.

The SET command provides a great many computational and logical capabilities.

In short, these commands permit you to perform many different manipulations on data-base files.

This section is for instructing you in how to use the data-base system. It provides detailed descriptions of the new commands and examples of how they are used.

INITIALIZATION

The data-base is not a separate program, but rather a group of commands. These commands are made up of the following files.

COMPLETE.TMK	SET.TMK
GROUP.TMK	SELECT.TMK
ORDER.TMK	UPDATE.TMK

SPECIAL CHARACTERS

The data-base commands make use of braces (i.e., { }), as do the LOAD, UNLOAD, AND TRANSFER commands. If you have changed these braces to other characters (using Option 2 of T/Modify), then the data-base commands will expect these other characters in places where normally they would expect to see braces. In other words, you would use your new characters everywhere braces are called for in the data-base system. Of course, you can instead use T/Modify to return to braces.

An Overview

T/Maker's data-base system begins with an ordinary text file. T/Maker permits the user to transform such a text file into a data-base file. Once transformed, T/Maker's new data-base commands provide a powerful array of utilities. In order to make this transformation the user must provide T/Maker with a model of the "card" or "record" that provides the basic structure for the data-base.

The Reference Section proceeds as follows:

I. Constructing the model "card" or "record."

The model record--or "Record Definition"--is the foundation upon which the data-base file is built. You don't use any of the new data-base commands to construct the Record Definition, though. You use only the T/Maker Editor.

II. Putting data into a data-base file.

Putting data into the data-base file is the job of the UPDATE commands.

III. Manipulating data and producing reports.

Manipulating data and producing reports are the jobs of the SELECT, ORDER, GROUP, SET, and COMPLETE commands which are often used in conjunction with standard T/Maker III commands like COMPUTE.

Establishing a Data-Base File: The EDIT Command

PRELIMINARY CONSIDERATIONS

It's convenient to begin with some vocabulary.

A "data-base" is a file (or a number of files) made up of "data records" (or "records" for short). A "record"--as we've already mentioned--is the computer equivalent of an "index card," which is to say, a specified set of information for a given case.

Consider the example of a city telephone directory. A telephone book on disk could be called a data-base. Each individual's entry in a telephone book is called a "record." Each record in a telephone book is made up of three items: a name, an address (optional), and a telephone number. Each of these three items--name, address, and number--is called a "field."

To use T/Maker's data-base capabilities, one must first set up a data-base file, just as one would set up any T/Maker file--using the T/Maker Editor.

After the file is created, you must provide T/Maker with a "Record Definition."

This format is typed into a file, starting a little below the file's top line. (T/Maker reports data-base messages on the file's top three lines, and so it is convenient to begin lower down.)

A "Record Definition" for a telephone book might look like this:

A Record Definition for a Telephone Directory

```
<record>
Name:      {name
Address:   {address
Telephone: {telephone
<end>
```

Let's examine the Record Definition more closely.

First, notice that it begins with "<record>"--this tells T/Maker that what it has here is not just any old text but a "Record Definition." The <record> can appear anywhere on the first line, and can be typed in capitals or small letters or a mix of both.

Next come three lines with "line-labels" to the left and "fields" to the right. The "line-labels" identify the lines of data. The braces identify the area where data will be placed, and the names inside the braces identify the specific data to be placed in each.

Finally, there is an "<end>" designation, telling T/Maker that the Record Definition has been completed. This can be anywhere on the line.

This Record Definition shows T/Maker an example of how the data in the data-base file should be stored.

Record Definitions may also provide the UPDATE command with the format you will be presented with when you want to update a data-base file. For example, when you are in the UPDATE mode, the telephone book's Record Definition (from the previous page) will appear on the screen as this:

The Screen Displayed by UPDATE for a Telephone Directory

Name:

Address:

Telephone:

As you can see, T/Maker provides you with the line labels and blank space to the right of the labels for filling in each person's particular data.

Thus, the Record Definition may provide both the format in which data will be stored and the template that is presented to the user when data are input or revised.

As it happens, from this small duality of purpose springs an important feature of T/Maker's data-base system. You see, these two functions are not entirely compatible with each other. When storing data for large files, one's main concern is economy, i.e., to store data in the least possible space. When inputting data, one's main concern is clarity, i.e., that line labels--which tell the user what data should go where--are clearly designated.

The incompatibility here is a simple one: Though it is a good idea to have good, descriptive line labels for the input of data, it is unnecessary and wasteful of space to have complete labels stored with each set of data permanently. Consequently, T/Maker permits you to use two different formats--one, the "Record Definition," for how you want the data stored, and the other, the "Form Definition," for how you want the form to appear on the screen when the data are being input.

A "Form Definition" for the telephone book might look like this:

A Form Definition for a Telephone Directory

```
<form>
    Name: {name}
    Address: {address}
    Telephone: {telephone}
<end>
```

The Form Definition begins with "<form>." Since we want this form to present clear and intelligible labels, this form has full labels for the lines of data in the record. Once more, names inside the braces show where data will be placed. As in the "Record Definition," the "Form Definition" ends with "<end>." The "Form Definition," then, looks like our original "Record Definition," save that the FD has <form> rather than <record> at its top.

The splitting of the storage function from the data-input function now permits us to be brutally cryptic in constructing the "Record Definition." All we need do, in fact, is provide the record definition a one-number (or one-letter) designator for each line of data, say like this:

A Concise Record Definition for a Telephone Directory

```
<record>
1 {name}
2 {address}
3 {telephone}
<end>
```

Obviously, this will save much storage space over the whole of the data-base file.

Both these templates--the Record Definition and the Form Definition--go at the top of a data-base file. The Form Definition always comes first. Thus, our exemplary data-base file should look like the following:

The Beginning of A Data-Base File for a Telephone Directory

```
<form>
    Name: {name
    Address: {address
    Telephone: {telephone
<end>
<record>
1 {name
2 {address
3 {telephone
<end>
```

Splitting the storage from the input functions in turn implies a series of refinements that apply to each sort of definition: Those concerning the "Record Definition" have primarily (but not exclusively) to do with storage economy, and those concerning the "Form Definition" have primarily to do with presentational matters. Let us begin with the Record Definition:

RECORD DEFINITIONS: VARIATIONS AND REFINEMENTS

A variety of options can be used in the design of "Record Definitions."

HANDLING MISSING DATA

Not all your records may have complete information. For example, consider the case of the telephone directory once more. Some telephone patrons won't want their addresses listed. That is to say, they will have "missing data" on the line for addresses. Naturally, it is a waste of space to store space for information that does not exist.

T/Maker will economize on space by simply dropping lines with no data. You avail yourself of this option by adding two right-braces ("double-braces") to the right of the line, like this:

A Record Definition for a Telephone Directory

```
<record>
1 {name
2 {address
3 {telephone
<end>
```

The double-braces on the address and telephone number lines will drop these lines from storage when their data are absent. It

is a good idea not to make the first line of data optional--so that T/Maker can always tell where a new record begins.

MULTIPLE FIELDS ON A LINE

T/Maker permits you to place more than one field on a single line. In general, T/Maker counts against your storage every character--blank or otherwise--out to the last nonblank character on a line (plus one byte). It is wise to use this principle in deciding between the relative advantages of one versus more-than-one field per line.

LONG NAMES IN SHORT FIELDS

An important method for economizing on space is to make field lengths no longer than necessary. One consideration in this regard is the length of the name of the field. Suppose, for instance, that it takes only one character (m or f) to record gender, but six characters to make "gender" the name of the field.

Long names can be placed in short fields by using the symbol "#" instead of a close brace to show where the brace would have been placed had the name been short enough (this is more fully explained in the reference section for the LOAD command). Even so, the shortest field definable in this manner is two characters wide because the braces themselves constitute and define useable storage space, example:

```
{#age}
```

To define a field only a single character wide, follow the number sign by another number sign, like this:

```
{##gender}
```

JUSTIFICATION OF FIELDS

T/Maker allows you to center, right-justify, or left-justify information in a field. When you generate reports using the data-base commands, you will want to control how the field's data are presented--say, for example, to make sure that numbers line up in a neat row. T/Maker also uses a field's justification as an indicator of whether that field contains alphabetic characters or numbers. As we will see later, this can be important.

You set a field's justification by introducing one of four symbols into the field somewhere between the left and right field braces:

FIELD JUSTIFICATION WITHIN RECORDS		
Justification	Symbol	Consequence
	<	Data will be left-justified
	>	Data will be right-justified
	-	Data will be centered
	!	Data will be "squeezed"

When a field contains no justification character, the justification character of the previous field is used. If you place no Justification symbols between field braces, T/Maker left-justifies all data.

Using the exclamation point (!) "squeezes" the presentation of fields, to compress the spaces between them. This option must be used with caution, however. It is ideal for some purposes: for instance, bringing the last name next to the first name for a mailing label. But once data have been squeezed in this manner they no longer appear in their proper columns for readability by the data-base commands. Thus, the exclamation point should be used only when data are being put into a text file to be presented in their final or reported form (this distinction will become clearer later).

PICKING FIELD NAMES

Each field in a record should have a unique name. Good names are pithy and memorable.

It's also a good idea to avoid field names that might appear as values in the field. For example, suppose that you gave the name "Jones" to a field that will contain surnames. Because "Jones" might appear both as a blank field's name and as a real value, this could prove troublesome for T/Maker's SELECT command.

Names should not begin with a number but rather with a letter.

Names should not include any of the following symbols: >, <, =, -, !, +, /, ", ', {, }, *, and #.

When a field has no name in it, the name is assumed to be "one higher than the last name used" (following the convention of the LOAD command).

A name may have a space in it when it is used within braces:

```
(last name )
```

This space must be removed when making a reference to the field for other purposes such as commands, example:

```
select address when lastname = Jones end
```

A colon or period makes a better separator than a space, since it can be used identically within braces and within commands.

Finally, THE CASE OF A FIELD NAME IS IMPORTANT--"last:name" is not the same name as "Last:Name."

THE <HERE> OPTION

In its ordinary operating mode, T/Maker begins filling up a data-base file with data immediately below the Record Definition.

For example, if we had input three telephone numbers into our telephone data-base our file would look just like this:

An Example of a data-base File for a Telephone Directory

```
<form>
Name:      {name } }
Address:   {address } }
Telephone: {telephone } }

<end>
<record>
1 {name } }
2 {address } }
3 {telephone } }

<end>
1 Robert Jones
2 123 Elm St.
3 344-7734
1 Doris Lange
2 5512 Oak St.
3 564-0987
1 Edward Magnus
2 191 Pine St.
3 446-4218
```

There are a variety of circumstances in which you may not want the data to appear immediately below the Record Definition. For example, you may simply want to have a heading appear above your data--something like what follows:

"Telephone Listings for Berkeley:"

In this case you can place a "<here>" on a line somewhere below the Record Definition. This "<here>" tells T/Maker to begin filling data from this point on down. A file using this option, then, would look like this:

An Example of a data-base File for a Telephone Directory

```
<form>
Name:      {name}
Address:   {address}
Telephone: {telephone}
<end>
<record>
1 {name
2 {address
3 {telephone
<end>
```

Telephone Listings for Berkeley:

```
<here>
1 Robert Jones
2 123 Elm St.
3 344-7734
1 Doris Lange
2 5512 Oak St.
3 564-0987
1 Edward Magnus
2 191 Pine St.
3 446-4218
```

As we will see, the <here> option can prove handy for a variety of purposes.

DATA-BASE FILES AS TABLES [1]

So far, we have thought of a record as the package of information for a given case. So far, also, our only consideration has been how to store that record. But the structure of a record may permit some other utilities as well.

[1] In this section, it is important that the user is familiar with the COMPUTE command. Please review the Spreadsheet part of the manual if you have never worked with COMPUTE before.

For example, you might want to set up your data-base file as a table. This lets you input data as it comes in, make calculations on your table when you desire, and even print your table when you want to.

Consider, for example, the data-base described below:

```
<form>
Product: {product      <> Price: {price  >}  Quantity: {quantity}>
<end>
<record>
+      {product      <>      {price  >} {quantity}>
<end>
ex          999,999.99      999,999      999,999.99
aci          *           +           =
              <here>
+      Widgets          45.98          35
+      Widget Refills   21.50          11
```

Here we see a data-base file into which the user inputs three pieces of information: the product's name, its price, and the quantity sold.

Notice that the `<here>` option has been used to start the data registering down below the example and calculation lines.

The user might use this sort of a data-base file to keep a running record of sales and revenues. After new data had been input, he would simply COMPUTE the table. This, of course, would serve to fill in the final column of figures, like this:

```
<form>
Product: {product      <> Price: {price  >}  Quantity: {quantity}>
<end>
<record>
+      {product      <>      {price  >} {quantity}>
<end>
ex          999,999.99      999,999      999,999.99
aci          *           +           =
              <here>
+      Widgets          45.98          35          1,609.30
+      Widget Refills   21.50          11          236.50
```

Incidentally, nothing stops the user from adding a variety of presentational embellishments to the file. Say, for example, one wanted a proper table title, headings, and a ".clean" command so that the table would be pruned of calculating symbols when it was

printed. The data-base file might then look something like this:

```
<form>
Product: {product      <> Price: {price  >}  Quantity: {quantity})
<end>
<record>
+      {product      <>      {price  >} {quantity})
<end>
.clean
=====
THE WIDGET PRODUCT LINE: RECORD OF SALES AND REVENUES
=====
Product:          Current Price:    Quantity:        Revenues:
-----
ex                999,999.99    999,999           999,999.99
acl               *             +                 =
..<here>
+      Widgets            45.98        35       1,609.30
+      Widget Refills     21.50        11       236.50
=====
```

The table arrangement above would be the appearance of the data-base file. When printed from the line containing ".clean," though, the table would come out looking like this:

```
=====
THE WIDGET PRODUCT LINE: RECORD OF SALES AND REVENUES
=====
Product:          Current Price:    Quantity:        Revenues:
-----
Widgets            45.98        35       1,609.30
Widget Refills     21.50        11       236.50
=====
```

MULTIPLE RECORD DEFINITIONS

It is also possible to employ more than one record definition in a single data-base file. Recall that the Record Definition shows T/Maker how to store data in the data-base file.

Why ever have more than one Record Definition?

As it happens, you can employ another Record Definition to change T/Maker's "unit of analysis." To explain:

T/Maker thinks about a data-base file on a record by record basis.

A record, in other words, is the "unit of analysis" or the "package" of information that T/Maker considers when it does data-base manipulations.

But there may be times when you want to look at your data-base file with a different unit of analysis in mind.

Consider an example:

Suppose you had a data-base file made up of all the basketball teams in the National Basketball Association. To keep matters simple, let's say that the file consisted of the team name followed by the first five players of each team and his field-goal and free-throw averages. So, the file (at least two teams worth of it!) would look something like this:

```
<record>
Team: {team <      }
Player: {player1 <    } fg avr: {fgav1 >} ft avr: {ftav1 >}
Player: {player2 <    } fg avr: {fgav2 >} ft avr: {ftav2 >}
Player: {player3 <    } fg avr: {fgav3 >} ft avr: {ftav3 >}
Player: {player4 <    } fg avr: {fgav4 >} ft avr: {ftav4 >}
Player: {player5 <    } fg avr: {fgav5 >} ft avr: {ftav5 >}
<end>
Team: Warriors
Player: Adams          fg avr:     .444 ft avr:     .678
Player: Baker          fg avr:     .398 ft avr:     .567
Player: Carter          fg avr:     .541 ft avr:     .703
Player: Debbs          fg avr:     .234 ft avr:     .411
Player: Edwards         fg avr:     .489 ft avr:     .675
Team: Knicks
Player: Andrews        fg avr:     .444 ft avr:     .378
Player: Birch           fg avr:     .358 ft avr:     .467
Player: Chase           fg avr:     .546 ft avr:     .603
Player: Daniels          fg avr:     .274 ft avr:     .511
Player: Ellington       fg avr:     .589 ft avr:     .775
```

Notice that the players within each team are organized into alphabetical order. Suppose, though, that--within teams--you wanted to organize the players according to field-goal percentages, in descending order. How to do it?

Well, in a manner of speaking, one can trick T/Maker into thinking that its unit of analysis is the player rather than the team. How? Just like this:

All you need do is add a second Record Definition to the file, this one just below the first record definition:

```

<record>
Team: {team <      }
Player: {player1 <    } fg avr: {fgav1 >} ft avr: {ftav1 >}
Player: {player2 <    } fg avr: {fgav2 >} ft avr: {ftav2 >}
Player: {player3 <    } fg avr: {fgav3 >} ft avr: {ftav3 >}
Player: {player4 <    } fg avr: {fgav4 >} ft avr: {ftav4 >}
Player: {player5 <    } fg avr: {fgav5 >} ft avr: {ftav5 >}
<end>
<record> PLAYER RECORD ONLY
Player: {player <    } fg avr: {fgav  >} ft avr: {ftav  >}
<end>

```

How will this new Record Definition aid us? Well, we need to introduce a couple of principles to understand the matter:

- (1) First of all: Keep in mind that T/Maker only employs one Record Definition at a time. Equally important, the Record Definition it employs is the first one in the file that it "sees" (the top line on the screen and down). (The Update command is the one exception; it always uses the first record definition in the whole file.)

Consequently, T/Maker will ordinarily work on the basis of the first and not the second Record Definition in this file of basketball teams.

- (2) It follows from the first point that if you quit the Editor while the cursor lies on the <record> line of the second record or use the Find command to advance the screen to this point, then T/Maker would never "see" the first Record Definition.

To T/Maker the data-base file would now look like this:

```

<record> PLAYER RECORD ONLY
Player: {player <    } fg avr: {fgav  >} ft avr: {ftav  >}
<end>
Team: Warriors
Player: Adams          fg avr:     .444 ft avr:     .678
Player: Baker          fg avr:     .398 ft avr:     .567
Player: Carter         fg avr:     .541 ft avr:     .703
Player: Debbs          fg avr:     .234 ft avr:     .411
Player: Edwards        fg avr:     .489 ft avr:     .675
Team: Knicks
Player: Andrews        fg avr:     .444 ft avr:     .378
Player: Birch           fg avr:     .358 ft avr:     .467
Player: Chase           fg avr:     .546 ft avr:     .603
Player: Daniels         fg avr:     .274 ft avr:     .511
Player: Ellington       fg avr:     .589 ft avr:     .775

```

And what does T/Maker see when it looks at this file this way?

Well, let us consider the matter. Remember that a Record Definition tells T/Maker what the lines look like. Anything that doesn't look like a line of a record, T/Maker will regard simply as a line of text--much the way we earlier saw that T/Maker permitted us to provide text in the form of headings and a border for a Table.

T/Maker will have no trouble "seeing" the player's lines as valid records. Their format is the same as the Record Definition.

But T/Maker will not "see" the "Team:" lines in the data-base file as data lines. It will see those two lines simply as text. So, T/Maker ignores those two lines. In fact, those two lines will be regarded much the same way as if they had been left blank.

That's important. It means that there is--in T/Maker's mind--a break between the Warriors' five players and the Knicks' five players. T/Maker's ORDER command will not order data across such a break.

Therefore, all we need do to order the players, within teams, by their field-goal percentages is give the following command while the cursor lies on the line containing "<record>" for the second Record Definition:

WHAT NEXT? order descending numbers fgav <return>

LINE LABELS

In the example for the previous section, the same line label (i.e., "Player:") was used for more than one line in the record. This practice cannot be used indiscriminately, since Line Labels play an important role in helping T/Maker identify where fields are placed within a record.

When T/Maker looks at records, it must decide which line in a file corresponds to which line in the record. This is accomplished by matching the Line Labels in the record against those found in the file. Look at the Record Definition below:

```
<record>
+    {amount}      ) {customer<           ) {invoice<}
-    {pay1>        ) })                   )
-    {pay2>        ) })
-    {pay3>        ) })
=    {balance      )
<end>
```

This record has been developed like a little table so that the file of records can be used with the Compute command to calculate balances. All the "pay-lines" start with a minus sign in the first column followed by six spaces. Thus, the Line Label for these is a minus sign and six spaces. All the "pay-lines" are also optional and may be missing if no data are present.

How then shall T/Maker tell "pay1" from "pay2" or "pay3" if only one "pay-line" is present? The answer is that it cannot. T/Maker matches the beginning of lines in the file against the Line Labels in the Record Description by assuming the next line in the file will probably be the one that comes immediately after the last line identified. Thus, after the first line has been found to start the new record, T/Maker will expect the next line to be the one representing "pay1".

If it were necessary to allow for a "pay3" or "pay2" without the presence of a "pay1," then the lines in the record would each have to start with a unique Line Label so that record processing commands could tell one "pay-line" from another (see below).

```
<record>
+      {amount}    ) {customer}          ) {invoice<}
-      P1{pay1}    ) })
-      P2{pay2}    ) })
-      P3{pay3}    ) })
=      {balance}  )
<end>
```

Note: Remember, T/Maker considers the "label" to be everything from the left margin to the first open brace. As with the Unload command, the length of the Line Label can be shortened without beginning the definition of the first field by placing a close brace at the end of the Line Label.

```
<record>
+ )           {amount}      )
<end>
```

The above defines a Line Label of just a plus sign in the first column.

THE FORM DEFINITION: VARIATIONS AND REFINEMENTS

TWO OR MORE FIELDS PER LINE

You may place more than one field on a single line. For example, you might wish an entry's first and last name to appear in two fields sited on the same line, like this:

```
<form>
  Name: {firstname}           } {lastname}
  Address: {address}          }
<end>
```

DRESSING UP THE FORM DEFINITION

You may want to dress up the Form Definition. For example, you may want the form to have a title, or a border, or you may want it to provide instructions on how it should be completed, like this:

```
<form>
  -----
  | TELEPHONE NUMBER FORM DEFINITION:
  |
  |           ----Enter last name, comma, first name
  |           |
  |           v
  | Name: {name}           }
  |
  |           ----Enter number, street, and city abbreviation
  |           |
  |           v
  | Address: {address}          }
  |
  |           ----Enter three digits, dash, four digits
  |           |
  |           v
  | Tel-number: {telephone }      }
  |
  -----
<end>
```

You supply all these embellishments with the T/Maker editor. They will appear as you designed them on the Form Definition. Do not, however, design a Form Definition larger than one screen-size, the maximum size limit.

MULTIPLE FORM DEFINITIONS

You may have more than one (and up to ten) Form Definitions per data set. Additional forms may be useful, for example, for handling two categories of sales, one sort requiring one sort of data and the other another sort.

All you need do is inscribe a second (or third, fourth, etc.) Form Definition below the first Form Definition at the top of the data-base file.

You select the Form Definition you want when you are in the UPDATE mode. Switch from one Form Definition to the next by touching the "Frame Mode" keystroke (this is "<ESC> F" in the stock T/Maker).

The "Frame Mode" keystroke is a "toggle option": This means that you proceed through the Form Definition options in series, ultimately returning again to the first option.

FIELD NAMES AND LENGTHS

Field names in a Form Definition must be spelled exactly as they appear in the Record Definition. The fields will be justified on the screen as they are justified in the record, thus no justification symbols need be given in the form.

The length of a field in a form will be the smaller of its length in the record and its length in the form. Thus it is a convenient practice to make the length of the field in the form greater than or equal to the length in the record.

In the case below, the UPDATE command would not permit more than one character to be entered for the field named "gender" even though eight columns are designated for it in the Form Definition.

```
<form>
Name: {name}                                } Gender: {gender}
<end>
<record>
1 {name}                                     } {##gender}
<end>
```

The UPDATE Command

DESCRIPTION

The UPDATE command is the heart of T/Maker's data-base system. You use this command to enter, revise, or delete data from your data-base file.

1/1 WHAT NEXT? UPDATE

UPDATE is not so much a command as a mode of operation, just as is the Edit command. In fact, you should think of the UPDATE mode as a special Editor designed specifically for data-base files.

Operating in the UPDATE mode lets you easily . . .

move the cursor from space to space, field to field,
record to record

delete or insert characters

send the cursor from one end of the file to the other

find data

add or delete records

temporarily store and transport records in the buffer

select among up to ten Form Definitions.

Once you have provided your data-base file with a Record Definition you open the door to the UPDATE command's various utilities.

The UPDATE command works on the working file, only.

Invoking the UPDATE command places an image of your record on the screen.

If you are building a new data-base file, the UPDATE command presents you with a blank record to fill in.

When you are working on a data-base file you have gotten from disk, invoking the UPDATE command presents you with the first record in the file.

Fundamentally, the command now lets you put data in the file or revise or delete data already in it.

Once you have made the inputs, revisions, or deletions you desired you must SAVE the file--once more, just as in T/Maker's Editor. If you do not SAVE the file, the changes you have made will not be permanently stored on disk.

When you invoke the UPDATE command you automatically transform your Editing keystrokes into ones specially useful for working with the data-base file.

UPDATE'S KEYSTROKES

The following charts show UPDATE'S new capabilities for the old Editing keystrokes. (Incidentally, if you use T/Modify to change a keystroke for the Editor, the corresponding keystroke for the UPDATE command is also changed.)

MOVING THE CURSOR WITHIN A RECORD

Intention:	Editor Keystroke To Use:
Cursor Left	Same
Cursor Right	Same
Move to the Field Immediately Below	Cursor Down
Move to the Field Immediately Above	Cursor Up
Advance to the Next Word	Same
Advance to the Next Field	Tab
Return to Previous Field	Set a Tab
RETURN	Returns cursor to first position of the field below. If the cursor lies on the final line, RETURN goes to the next record.
Go to First Field of the Record on the screen	Home the Cursor

=====

MOVING THE CURSOR FROM RECORD TO RECORD

=====

Intention:	Editor Keystroke To Use:
Go to Next Record (1st field)	RETURN the Screen
Go to First Record (1st field)	Home Screen
Go to Next Record (same field as presently)	Screen Down
Go to Previous Record	Screen Up
Go to First Record	Move to First Line
Go to Final Record	Move to Bottom Line

=====

=====

REVISING A RECORD

=====

Intention:	Editor Keystroke To Use:
Delete a Character	Same
Insert a Character	Same
Delete spaces to the Next Character	Same (deletes to field's end)
Erase to end of field	Erase to the end of line
Destructive Backspace	Same
Delete line	Same (deletes all fields on a line)
Insert Mode	Same

=====

=====

INSERTING AND DELETING RECORDS

=====

Intention:	Editor Keystroke To Use:
Delete a Record	Join Lines
Insert a Blank Record	Break Lines

=====

To add records to the end of a file, do the keystrokes for "Go to final record" followed by the keystrokes for "Go to next record."

<u>MOVING OR DUPLICATING RECORDS</u>	
Intention:	Editor Keystroke To Use:
Clear buffer	Same (the buffer holds only one record at a time and must be cleared before a new record can be placed in it or when changing forms)
Copy to buffer	Same (copies the complete record)
Move to buffer	Same (moves the complete record)
Insert buffer	Same (inserts the record ahead of the one currently displayed)

<u>FINDING SPECIFIC RECORDS:</u>	
Intention:	Editor Keystroke To Use:
Find a Record	Search for a String (specify a string appearing in the record)
Find Another Record containing the string	Continue Search

Searching for a string finds the first following record that contains the string anywhere in the record. You may want, for example, to search for a record that has an Invoice Number of 23, but will find instead a record that has an amount of \$23.00.

There is one approach which may be helpful. Continuing with the example above, suppose the Record Definition for the field "invoice" appeared somewhere on a line as shown below.

```
{invoice>}N
```

In this case, one could avoid happening on the record with an amount of \$23.00 by searching for the string "23N" even though the "N" is not a part of the field. If the invoice number were left-justified, then the "N" should be put to the left of the field and the string to specify would be "N23."

OTHER	
Intention:	Editor Keystroke To Use:
Change form	Frame Mode (may not be used when the buffer is full)
Quit Update	Quit the Editor
Repeat last keystrokes	Same
Redraw the screen	Same
Information	Same (provides file name and the approximate space remaining)

THE "CHANGED" FIELD

T/Maker provides another option within UPDATE which automatically contributes a new and useful field to the record. This is called the "CHANGED" field. The "CHANGED" field serves to inform the user if, over the course of updating a given data-base file, a given record has been worked upon.

This option, in other words, is a bit like a T/Maker fingerprint test on records--if you have touched a record in a work session, that record's CHANGED field will register a "1" in it; if that record was left untouched, then the CHANGED field will register a blank.

You avail yourself of this option simply by installing a changed field in your Record Definition. Since this field will contain at most a one-character value (that is, the "1" it contains when the field has been touched) we recommend that you use the method described previously for establishing a one-column field. Hence, your changed field should look like either this,

(##CHANGED)

or, this,

(##changed)

on some line in your record. The word "changed" must appear in all-caps or all-smalls.

It bears noting that merely bringing a record to the screen or passing the cursor over it will not count as "work" (whatever your own views on the matter). If, however, a number or letter is

keyed-in to the record, the "changed" field will go to "1."

The field is particularly useful in conjunction with the SET command which can be used to return the "changed" field to spaces. Some reasons for using this option are discussed in the section on the SET command that deals with conditionals.

The SET Command

DESCRIPTION

The SET command allows you to derive new data from the existing stock of data in a record. The newly derived data are lodged into the record.

This capability is quite powerful. For example, the SET command can perform numeric computations to generate new data. The results of such calculations are in turn lodged into the record. Suppose your data-base record happened to include data on a product's price and quantities sold. The SET command would happily calculate a figure for total revenues from these two figures.

The SET command will also perform logical operations based on alphabetic character strings. For example, suppose you wished to create a field reflecting four categories of persons: single males, single females, nonsingle males, and nonsingle females. Say your record presently contains information on both gender and marital status but you want the information combined in a single, new field. Once more, the SET command will easily perform this task.

Indeed, numerical and logical capabilities can be combined together. For instance, a numerical quantity may be calculated for every record and then a new, dichotomous variable created depending on whether the calculated figure fell above or below a certain criterion value.

WHAT NEXT? set

The SET command does its job in a convenient way. Perhaps its operation will be easiest to understand if you will again think for a moment of the index-card model of a data-base file that we spoke of in this manual's introduction.

Imagine, if you will, that you possessed a magical card-file box containing magical index cards. These index-cards did not merely sit passively in their box. Their magical property lay in the fact that they could do work on the information the cards contained. All you had to do was write some instructions on a card, and the card would carry out the task you requested!

Well, this is very much what the SET command does. To request that SET make a calculation or perform a logical operation, you must show the record what thing you want it to do.

This is easily accomplished.

RULE DEFINITIONS

Recall that the top of the data-base form is where we put both the Form Definition and the Record Definition. This is also the territory where we shall want to place the instructions that tell the SET command what to do--such an instruction is called a "Rules Definition".

The SET command is always instructed by giving it one or more RULES to follow. You provide a rule that defines the result or results you want to generate. You will also have to modify the Record Definition so that it includes a field into which the results can be placed in the record. When these two things--the RULE has been provided, and so has a field for the result--are done, you invoke the SET command. Like the COMPUTE command, the SET command quietly does the work you've requested. The calculation or logical exercise you've asked for will be carried out on all records--or, at least, all that have the required data.

Just as the Record and Form Definitions did, the Rules Definition begins and ends with special lines. The beginning line is "<rules>" and the end line is "<end>".

The rules for SET to follow are placed between the beginning and end lines.

A Rules Definition, like a Form Definition, is always sited ahead of the Record Definition. Let us look at one example:

```
<rules>
total = price * quantity
<end>

<record>
1 {quantity} {price} {total}
<end>

1          10        10
1          15        20
```

In this example, the rule provided defines a new field called "total." "total," the rule says, is defined as the product of the values in two already existing fields, "quantity" and "price." Notice that the Record Definition provides a field for "total." Otherwise, our result would have no place to go.

REQUESTING MATHEMATICAL CALCULATIONS WITH RULES

RULES requesting a mathematical calculation should always be expressed in a certain way.

A calculation is always written on a single line--though you may extend that line up to 300 or 400 characters depending on your computer. Upon this line, first comes the name defining the result you want. Then comes an equal sign. Then comes the formula that defines the desired calculation.

You can use any of the following in your RULE formula: parentheses, numerical constants, field names, and operators + for addition, - subtraction, * multiplication, / division.

For example, suppose that we wanted to calculate the sales tax associated with a particular product. The tax itself, let us say, is a constant, 6%. The tax applies to revenues from that product, which are calculated as the quantity times the price. This quantity might be enclosed in parentheses (though this is not, of course, necessary) for clarity's sake. Here is how our Rule Definition would look:

```
<rules>
    tax = 0.06 * (quantity * price)
<end>
```

A slightly more complicated calculation might look like this:

```
<record>
    ratio = (1 + labor)/(factor1 * (factor2 - factor3)) end A-T-RATIO
<end>
```

Above we see a calculation that requires the use of parentheses in order to make clear the right order of calculation. Notice, incidentally the use of the word "end" in this RULE. The word "end" can be used to tell T/Maker that the formula for the calculation has come to an end. Space after the "end" word can be used, for example, to give a title to the Rule--in this case that title is "A-T-RATIO."

MODEL NUMBERS: Sometimes it will be desirable to specify how the result of a given calculation should be presented--for example, you may want to specify how many decimal places of accuracy the result should be carried out to. This can be accomplished by placing a "model number" (see the COMPUTE command) immediately after the name of the result field. For example:

```
<rules>
    tax 999,999.99 = .06 * (quantity * price)
<end>
```

When you don't use a model number T/Maker presents the result carried out to as many significant figures as necessary, to a maximum of 13. If such a number will not fit in the field it is assigned, T/Maker truncates the number on the right. So, for example, 12.64527695964 fitted into a 7-column field would appear as 12.64527. Clearly, it's a good idea to provide model numbers whenever your calculation may produce long and unwieldy results.

You may also use model numbers as a means for reformatting existing data in your record. Suppose, for example, that in your current record the price variable were expressed simply in dollars (i.e., no cents) but you wanted to change the presentation to include cents. By putting that variable's name on both sides of the equal sign and giving a model number to the left side you can change the number's presentation.

```
<rules>
    price 999.99 = price
<end>
```

ZERO VALUES: Like the COMPUTE command, you may also specify how values that fall on zero exactly should be presented. You do this by following the model number with quotation marks that enclose the desired zero-values string. If you wish the zero-values string to be a blank, enclose a blank in quotation marks. As in COMPUTE, values that are not exactly zero but would be correctly reported as all zeroes according to the model number in use will be presented according to the model number rather than the zero-values string.

In the example below a price of nil will be presented as blank; a total of nil will be presented as the characters "No Charge."

```
price 999.99 " " = price
total 99,999.99 "No Charge" = price * quantity
```

SEQUENTIALLY NUMBERING RECORDS

Now and then you may find that you would like to have your records sequentially numbered--that is to say, numbered according to their succession in the data-base file.

T/Maker's SET command provides a special option for doing just this.

The new sequence number will be a variable just like any other, and so it must be provided with its own field. You may name this field whatever you like (within the limits for field

names described earlier). Let's call our sequence-number field "id" for the moment.

The rule you provide for getting T/Maker to supply sequential numbers in this field is simply this:

```
<rules>
  id = number
<end>
```

As it happens you may write "number" as "number" or "NUMBER" but not any other way (for example, "Number"). This simple rule, then, will put sequential record numbers into the field called "id" when the SET command is invoked.

The number option may also be used as a value in mathematical formulas. For example,

```
col 999.99 = number / 30
```

USING SET TO MANIPULATE ALPHABETIC CHARACTERS

You can use the SET command to manipulate alphabetic characters in a couple of ways: (1) SET can assign a specified character string to a field; (2) SET can also concatenate two or more strings into a single, new string.

ASSIGNING A CHARACTER STRING: You can use the SET command to supply a specified character string to all records.

Suppose, for example, that you were preparing the sales data for the state of New Hampshire. You want the State's name to appear on every record, but you want to save yourself the trouble of actually having to key in "New Hampshire" so many times. Well, you can use the SET command to supply a new field called "state" with the string "New Hampshire."

The Rule Definition would look like this:

```
<rules>
  state = "New Hampshire"
<end>
```

T/Maker sees this rule as saying: "In the field called 'state' put the character string (you can think of it as a 'result' if you like.) 'New Hampshire.'"

When the string you want to supply to records contains a blank or a nonalphabetic character the string should be enclosed in single or double quotation marks.

CONCATENATING STRINGS: Placing an exclamation point between two strings will concatenate those strings in the resulting field. For example, the rule below will place the string "abcdef" in the field named "test."

```
test = "abc" ! "def"
```

APPLYING RULES UNDER CERTAIN CONDITIONS

So far, the Rule Definitions we have described have applied to all of the records in a data-base file.

T/Maker also permits you to apply a Rule selectively. Say, for example, you have a data-base file comprised of the sales records of salespeople. You want to use the SET command to compute bonuses for them. But you want only those salespeople who generated revenues of more than \$100,000 to get bonuses.

In this case, you will want to establish a condition under which a bonus calculation is done. T/Maker makes it easy to introduce such conditions into a Rule Definition. You simply add the word "WHEN" at the end of the calculation formula and then add your specifications for the condition that should apply. For example, our bonus calculation might now be specified thus:

```
bonus = .01 * revenues when revenues > 100,000
```

OPERATORS AVAILABLE FOR USE IN CONDITIONAL CLAUSES: Your specification of a conditional clause can include the use of field names, constants, mathematical operators, or parentheses.

EXAMPLE: Suppose money were tight and you decided to give that bonus only when a salesperson's NET revenues exceeded \$100,000:

```
bonus = .01 * revenues when (revenues - expenses) > 100,000
```

You can use the logical words "and," "or," or "not" as operators. T/Maker's "or," incidentally, is inclusive, which is to say that the statement "A or B" is true if A is true, B is true, or both are true.

EXAMPLE: Suppose money were even tighter and you decided to give the bonus when net revenues exceeded \$100,000 AND his expenses were less than \$10,000. (Note: the rule

would be placed on one line, and not as shown below.)

```
bonus = .01 * revenues when ((revenues - expenses) > 100,000) AND  
                           (expenses < 10,000)
```

You can also use a variety of "comparison operators" specially provided to make comparisons in conditional clauses. T/Maker provides the following comparison operators:

Operator	Interpretation
<	less than
<=	less than or equal to
=	equal to
<>	not equal to (or, less than or greater than)
>=	greater than or equal to
>	greater than

Finally T/Maker provides three operators that specify the "relations of possession" among character strings.

The word contains requires that string-1 contains string-2. For example, string-1 "abcd" can be said to contain string-2 "bc".

The word within requires that string-1 can be found within string-2. For example, string-1 "bc" is within string-2 "abcd".

The word among requires that string-1 can be found in freestanding form in string-2. For example, string-1 "hot" is among string-2 "hot warm cold".

Note: "hot" is not among "hotter" or "hotel" even though these contain the string "hot".

EXAMPLE: Suppose you also wanted to give everyone in the Computer and Shipping departments a minimum of \$100, regardless of net revenues.

```
<rules>  
bonus = .01 * revenues when ((revenues - expenses) > 100,000) and  
                           (expenses < 10,000)  
bonus = 100 when bonus < 100 and department among "Shipping Computer"  
<end>
```

Let us examine these two Rule Definitions for a moment. Notice, first, that the upper rule is the one that carries out the bonus calculation for salespeople.

Look at the lower rule. Here we see T/Maker being asked to set the amount in the bonus field equal to 100 in records if the amount presently in the bonus field is less than 100 (a specification that excludes salespeople with higher bonuses) and when the employee's data in his "department" field is either "Shipping" or "Computer." The use of the "among" operator in this case provides the means for specifying department membership. This last condition will, of course, exclude all members of the Sales department (or other non-Shipping or non-Computer departments) from the bonus calculation.

Keep in mind that T/Maker goes about its SET tasks progressing record-by-record and rule-by-rule through the records. In other words, the work demanded in one rule is completed before the work of a second rule is begun.

POTENTIAL RISK OF CONFUSION BETWEEN NUMBERS AND ALPHABETIC CHARACTERS: When two values are compared using one of the comparison operators described above, the values can be compared either as alphabetic characters or as numbers. Sometimes, this affects the results of the comparison. For example, as far as numbers go 0.0 is equal to 0, but as far as alphabetic characters go 0.0 is not identical to 0. [1]

You may want to make it explicit to T/Maker that you wish an alphabetic or a numeric comparison in a given instance. To do this, put the word "char" in front of the comparison operator for a comparison of alphabetic characters. Use the word "numeric" in the same way for a comparison of numeric values.

[1] This sort of confusion between numbers and letters can have dramatic consequences. For example, the number "30" is, of course, greater than the number "4." However, as alphabetic strings T/Maker sees "30" as alphabetically preceding (and, thus, effectively "less than") "4." This is so because T/Maker compares alphabetic character strings starting with the left-most nonblank character. Thus, in this case, the "3" in "30" is compared to the "4" in "4," making "30" less than "4."

The SET command will ordinarily make a numeric comparison. There are two exceptions to this rule. SET will make an alphabetic comparison when:

any one of the fields in the Rule Definition is right-justified or centered, or

a constant is enclosed in quotation marks or does not begin with a digit, period-digit, or minus sign-period-digit, or minus sign-digit.

If you generally right-justify numbers and center or left-justify text strings, the SET command will nearly always pick the correct type of test.

MIXING MATHEMATICAL, ALPHABETIC, AND LOGICAL OPERATORS

T/Maker allows you to mix mathematical, alphabetic, and logical operators.

Whenever a logical test is involved in an expression, the result must either be that the conditional is true or it is false. T/Maker expresses this result as a "1" for "true" and a "0" for "false." [1]

The 1's and 0's that are produced as a result of logical operations can, in turn, be manipulated as digits if you like. For example, a teacher might use the expression below in order to sum the number of questions that students answered:

```
answers = (a1 <> " ") + (a2 <> " ") + (a3 <> " ")
```

Note: The quotation marks enclosing blanks in the above expression denote that a student's datum in the specified field is missing. Blanks enclosed by quotation marks means "missing data." Blanks enclosed by quotation marks can be used more generally when one wants to specify the handling of missing data in an expression.

[1] In fact, a value itself is true if it converts to a nonzero number, thus the two rules below are identical.

```
action = NOTIFY when balance numeric <> 0  
action = NOTIFY when balance
```

T/MAKER'S SEQUENCE OF PROCESSING OPERATORS: You can always use parentheses to make the sequence of calculations and operations clear and explicit to T/Maker.

If you do not provide parentheses, however, T/Maker follows its own, pre-set order of sequence of processing through an expression. It may be useful for you to know this sequence--for one thing, such knowledge permits you to dispense with parentheses, thereby speeding up your work.

T/Maker processes an expression by looking first for * (multiplication) or / (division) operators--these are calculated before anything else. The order of succession for remaining operators is described below:

T/MAKER'S
DEFAULT PROCESSING SEQUENCE

* and /	done first
+ and -	done second
!	done third
< = etc.	done fourth
not	done fifth
and	done sixth
or	done seventh

Thus, for example, T/Maker will process the expression,

a + b * c < 5 or d = 5 and c >= 1

as,

((a + (b * c)) < 5) or ((d = 5) and (c >= 1))

Of course, parentheses can be--and always should be--used to make sequencing explicit in more complex expressions. T/Maker, incidentally, permits you to nest pairs of parentheses inside other pairs of parentheses, as the example immediately above suggests.

HOW TO APPLY A CONDITIONAL EXPRESSION TO A GROUP OF RULES

T/Maker provides a shortcut device for applying the same conditional expression to two or more Rule Definitions.

This is done by writing the conditional expression you wish to apply, by itself, on a line that begins with the word "when." Such a line is called a "general conditional." Rule Definitions

on all lines below such a "general conditional" line will follow that conditional. Such a conditional expression is superceded when a new "general conditional" is provided.

In the example below, general conditionals are employed in the Rule Definition on the left side to accomplish the same task that is accomplished on the right side without general conditionals.

CALCULATING THE TAX ASSOCIATED WITH SALES . . .

INCORPORATING GENERAL CONDITIONALS

```
<rules>
value = price * quantity
when type = free
  tax = 0
  total = value
when taxtype <> free
  tax = .06 * value
  total = value + tax
<end>
```

NOT INCORPORATING GENERAL CONDITIONALS

```
<rules>
value = price * quantity
tax = 0 when taxtype = free
total = value when taxtype = free
tax = .06 * value when type <> free
total = value + tax when type <> free
<end>
```

You may employ conventional conditional expressions in the rules that appear below a general conditional. In such a case, both the general and conventional condition must be true for the rule to be evaluated.

```
when department = Computer
  bonus = 100 when category = A
```

A general conditional can be cancelled for all expressions appearing below it by providing a line with a simple, true conditional--for example, "when 1 < 2".

USING "CHANGED" GENERAL CONDITIONALS TO SAVE TIME:
Processing SET commands that involve many records and many rules can take much time. Sometimes you can economize on processing time by incorporating the "CHANGED" field into a general conditional. (Remember that a record's "CHANGED" field is set to "1" [or, effectively, "true"] by the UPDATE command when it is first entered or changed.)

Suppose, for example, that you maintained a record of company invoices on a data-base file. You would save processing time by telling T/Maker to process only those invoices newly entered or changed. Such an instruction might look like the following:

```
<rules>
when CHANGED

rule-1
rule-2
etc.

CHANGED = " "
<end>
```

Notice that the last rule serves to set the "CHANGED" field to blank (i.e., effectively "false" or "unchanged"). This insures that subsequent uses of the SET command on the file will not reprocess any record unless the UPDATE command had first been used to change it.

The ORDER Command

DESCRIPTION

The ORDER command puts records into alphabetical or numerical order.

WHAT NEXT? order [ascending or descending] [numbers or characters]
field-name

You must specify which field the records should be ORDERED by. You can ORDER by any field in the record.

ORDER works only on the working file--to be exact, it works on the working area. Thus, it is possible to order only part of a file.

You can order in the ascending (T/Maker's default) or descending direction. To request "descending" order simply add the word "descending" between the order command and the field-name in the command string, like this:

WHAT NEXT? order descending field-name

You may wish to make explicit that the ordering to be done is either numeric or alphabetic. To do so simply add either the word "numbers" or the word "characters" (for alphabetic characters) to the command string between the order command and the field-name, like this:

WHAT NEXT? order numbers field-name

or,

WHAT NEXT? order characters field-name

These four option words--"numeric," "characters," "descending," and "ascending"--may be abbreviated in the command string. The only limitation on this abbreviation is that the abbreviated option word must not be subject to confusion with a field-name. So, for instance, if a record contained a field named "d" it would be necessary to spell out at least as much as "de" if one wished an ordering in the descending direction.

If you do not tell T/Maker which it is ordering, numbers or alphabetic characters, T/Maker takes its cue from the justification it finds in the field you are ordering by. It assumes that a right-justified field contains numbers. It assumes that either a left-justified or centered field contains alphabetic characters. Numeric orderings of centered fields will ignore column positions and order instead on the basis of the numbers' values.

When doing multiple orderings of a file, order on the least important field first, then the next-most important, etc.

The ORDER command will not order across non-record lines in the data-base file. In other words, it works on contiguous blocks of records only.

TIPS

Ordering a large file can take a lot of time. As it happens, no T/Maker data-base commands demand that a data-base file is ordered. Therefore time-consuming ordering tasks should probably be reserved for generating reports.

If ordering is frequently to be done on the basis of a number of fields, you may find it useful to invoke the help of the SET command. You may be able, for example, to use the SET command to construct a single field for use solely as a convenient ordering device. Consider the example of ordering a data-base file by date--supposing that our date information were entered as three fields: month, day, and year.

Ordering by date in the conventional way would oblige us to do three separate sorts--first by day, then by month, and finally by year. But it is an easy matter to employ the SET command to generate a single number--based on the date information--which could serve as the basis for date-order.

All we need do is create a new datum which is the sum of the following operations: the year multiplied by 10,000, the month multiplied by 100, and the day. Or:

```
<rules>
  date = year * 10,000 + month * 100 + day
<end>
```

By this method, then, 1/1/83 becomes 830101, 1/2/83 becomes 830102, and 2/1/83 becomes 830201--these dates would be Ordered appropriately by ordering on the new "date" field.

The Group Command

The GROUP command summarizes data in a data-base file and presents a report of its results.

The working file will be changed as a result of using the GROUP command-- hence, one should take care to RENAME the new file if one wants to retain the original data-base file.

WHAT NEXT? group [list] [key-field1, key-field2, etc.] end

This command provides a "summary record" based on one or more records for the same entity. In other words, GROUP provides for a "collapse" of data. This collapse can be said to involve two aspects, a weeding aspect and an additive aspect.

The GROUP command "weeds out" records by removing multiple records for the same entity, leaving a single representative record of that entity behind. The GROUP command works additively in that it sums up numerical data drawn from the same field across multiple records for a single entity; it places that sum in the one remaining record for that entity.

The GROUP command weeds out records according to the "key field" specifications that you provide.

GROUP looks for all identical instances in one or more key fields. Records that are identical in terms of key fields will be collapsed together into one summary record.

GROUP does one of two things with the other (non-key) fields in a record, depending on the kind of justification it finds in the field: Right-justified fields are arithmetically added together. Left-justified or centered fields are given the datum that appears in the first record's field for a given entity that it encounters.

T/Maker always regards the contents of key fields to be alphabetic--thus 1.0 is not the same as +1 in a key field.

All the records in a file can be combined into a single record by providing no key-field specification to the GROUP command (i.e., "GROUP END"). This may be useful in getting sums across all the records of a data-base file.

The GROUP command also puts three lines of "GROUP MESSAGES" on the screen at the completion of its function. These lines report the number of records found in the data-base and the number of records remaining after the GROUP command completed its work.

THE "LIST" OPTION: The "list" option is a memory jogger. Including the word "list" in the GROUP command will cause T/Maker to respond by presenting a roster comprised of the field-names present in the working file. This is useful, of course, if you happen to have forgotten the working file's field names or if you are unsure of their exact spelling.

It is necessary that you touch the RETURN key after requesting the "list" option. Touching RETURN brings the roster of field names to the top of the screen. T/Maker will now prompt you for the remaining specifications of your GROUP command. The prompt looks like this:

ON WHAT FIELDS (OR 'LIST' OR 'END')?

This prompt invites you to continue to provide the specifications of your GROUP command string once the field names have been presented at the top of the screen.

COUNTING RECORDS: The GROUP command does not provide a count of the number of records that may have been collapsed into a single entity in its report. There is a way to achieve such a count, however. All you need do is introduce a special field called "count" into the Record Definition.

(count)

When the Record Definition contains a field called "count," T/Maker automatically presents a count of the number of records that were collapsed into the representative record reported in the GROUP command's report. The total count is presented in the summary record's count field.

T/Maker accomplishes this end by automatically setting the contents of a blank count-field to "1" and then summing the numbers it finds in these fields.

This procedure allows the user to manipulate or manually override T/Maker's count. If, for example, you wished a particular record to have the value of "2" rather than "1" in a count of records, then all you need do is supply a "2" to the count field of that record.

AN EXAMPLE: Consider an example of the GROUP command's operation.

Suppose that the following data-base were a record of recent sales:

A RECORD OF RECENT COMPANY SALES

```
<record>
1 {Customer< } {City< } {product1>} {product2>} {product3>} {count>}
<end>
1 Customer A      City X          20        40        120
1 Customer B      City Y          30        60        20
1 Customer C      City X          20        50        90
1 Customer A      City Y          30        80        10
1 Customer B      City Y          180       50        120
1 Customer A      City X          40        80        20
1 Customer C      City X          120       30        70
```

If we wished to see these sales summarized by customer, we would give the following command:

WHAT NEXT? group Customer end

Notice, also, that the GROUP command statement must provide an exact version of the field name--in this case "Customer" and not "customer" or "CUSTOMER."

The GROUP command will in this case generate the following report in the working file:

THE GROUP COMMAND'S REPORT:

```
<record>
1 {Customer< } {City< } {product1>} {product2>} {product3>} {count>}
<end>
1 Customer A      City X          90        200       150       3
1 Customer B      City Y          210       110       140       2
1 Customer C      City X          140       80        160       2
```

The above report shows that three customers--A, B, and C--provided company sales. Quantities for each product have been summed for each of these customers. The "count" variable shows the number of separate records that went into each customer's summary report.

Notice that Customer A's City field is reported as City X in the GROUP'S summary report even though one of the sales to this customer was actually made in City Y (see the 4th record in the file, above). This is so because the GROUP command preserves the

first datum it finds in an alphabetic field.

A grouping by cities is gotten with the following command:

WHAT NEXT? group City end

In this case, the report will be this:

THE GROUP COMMAND'S REPORT:

```
<record>
1 {Customer<      } {City< } {product1>} {product2>} {product3>} {count>}
<end>
1 Customer A      City X          200        200        300        4
1 Customer B      City Y          240        190        150        3
```

A grouping by customers and cities is gotten with the following command:

WHAT NEXT? group Customer City end

In this case, the report will be this:

THE GROUP COMMAND'S REPORT:

```
<record>
1 {Customer<      } {City< } {product1>} {product2>} {product3>} {count>}
<end>
1 Customer A      City X          60         120        140        2
1 Customer B      City Y          210        110        140        2
1 Customer C      City X          140        80         160        2
1 Customer A      City Y          30         80         10         1
```

Notice that in this case Customer A's purchase in City X and City Y are presented separately.

TIPS: When a count field is not present in the Record Definition and all the fields in a record are left-justified or centered the effect of invoking the GROUP command is simply to remove records from a file. The first record in the file for any key field will remain intact while all others for that key field will be dropped. This feature can be useful, for example, in reducing a list of all orders by customer to a list of active customers per se.

The SELECT Command

The SELECT command selects or culls subsets of data from a data-base file. The SELECT command also reformats data to your liking.

The SELECT command permits two kinds of selections. You can cull either (1) particular subsets of RECORDS or (2) particular subsets of FIELDS (or both records and fields simultaneously).

This command is ideal for generating mailing labels from a data-base file, and it has many other uses as well.

```
WHAT NEXT? select {"it" or [disk-file-name]} {list}
               {from [record-number]} {to [record-number]}
               {when [logical-test]} end
```

SELECTING SUBSETS OF RECORDS

A SELECT command line properly consists of the word "select" followed by a number of command-words terminated by the word "end". These command-words specify important aspects of the selection to be done. Some of these command-words are optional and some are not optional. Let us consider them:

THE SOURCE OF SELECTED RECORDS

The first option of SELECT designates the "source" file--that is, the file FROM WHICH records should be culled. This command-word can legitimately be either "it" or the name of a file on disk.

"It" tells T/Maker to use THE WORKING FILE as the source file.

A disk-file name tells T/Maker to use THE FILE ON DISK NAMED as the source file.

If you employ the "it" option, the working file will be lost. It will be replaced by the post-culled data. For this reason it is often a good idea to SAVE and then RENAME the working file before employing this option.

When selecting from a disk file into an empty working file, the source file's Record Definition is reproduced at the top of the newly created file, and the format of the selected data remains unchanged. The selected records are placed in the working

file.

THE LIST OPTION

This word is an optional memory jogger. Putting "list" into the command-string at this point causes T/Maker to respond by presenting a roster comprised of the field-names present in the source file. This is useful, of course, if you happen to have forgotten the source file's field names or if you are unsure of their exact spelling and intend to use them in expressing a logical condition for selection.

It is necessary that you touch the RETURN key after requesting the "list" option. Touching RETURN brings the roster of field names to the top of the screen. T/Maker will then prompt you for the remaining specifications of your SELECT command.

FROM AND TO OPTIONS

FROM-TO specifications are used to tell T/Maker to cull all records from a particular record number to a particular record number (inclusive) in the data-base file. When neither FROM nor TO is specified, all the records in the source file are potentially included in the resulting file.

An example use of these options might look something like this:

WHAT NEXT? select data.1 from 1 to 100 end

In this command-line T/Maker is being instructed to cull records from a disk-file named "data.1." The FROM-TO specifications tell T/Maker to cull all records from the 1st record to the 100th record (inclusive) in the data-base file. The records need not be sequentially numbered in order to use the FROM-TO option--indeed, this option ignores the sequential numbers your records may have, working instead from the de facto order of records it finds in the file.

It bears mentioning that this option is particularly useful in a special case. You may now and then find that you wish to reformat data by means of the SELECT command (discussed later). The new format you desire may require more memory space than the old format, and your source file may be full or nearly full in terms of memory consumed. In this special case, then, you would use the FROM-TO option to distribute records to more than one new file, thereby providing the necessary memory space.

CONDITIONAL SELECTIONS OF RECORDS

A final option may be used to provide T/Maker with SELECTION CRITERIA for the culling you desire to do.

A conditional selection is optional in the sense that if you provide no culling criteria T/Maker will select ALL of the records in the source file.

If you wish to specify selection criteria, this must be the last option specified. You express this option by entering the word "when" followed by the words specifying the condition. The rules for this logical condition, and the words and symbols it contains, are described in the SET command's description in this manual. Please refer to that section for details.

Here are some examples of SELECT command lines incorporating conditional expressions:

```
WHAT NEXT? select data.1 when price > 100 or quantity > 25 end
WHAT NEXT? select data.1 when state = California end
WHAT NEXT? select data.1 when cartoon = "Tom and Jerry" end
WHAT NEXT? select data.1 when (3 * age) + weight > 300 end
```

REFORMATTING FIELDS WITH SELECT

So far, we have shown how SELECT can be used to cull records from a data-base file.

The format of records can be changed by the SELECT command, too. For example, you may wish the newly selected file to include a subset of the fields available in the source file, or you may wish to move a record's fields into new positions for a new appearance.

You can both reformat data and cull subsets of records in the same SELECT command.

Reformatting is done by providing a newly designed Record Definition at the top of the working file that receives the SELECTED data.

When the working area does not contain a Record Definition the Record Definition on the disk file is reproduced in working area. (The disk file, of course, must contain a Record Definition.)

For example, suppose you wished to reformat the fields of data in a list of employees in order to create an intra-office mailing label. Say that you were SELECTING from an employee list that is held on a disk-file called empl.1. It might look like this:

```
<record>
    (first name <   ) (last name     ) (room) (telephone)(>age)
<end>
=====
First Name           Last Name        Room   Telephone   Age
-----
<here>
Alexander          Gilbert         X-21   377-1830   30
Bob                Jones          M-45   377-1834   40
Richard            Allison        M-22   377-1901   62
```

Notice, incidentally, that this file makes use of non-record lines in order to provide headings for the employee list. The <here> optional line serves to send updated data to the area in the file below the headings.

We wish to draw our mailing-list data from this file.

How to do this?

First, we must create a blank working file with the new mailing label's appropriate Record Definition at its top.

Consider the record below:

```
<record>
.new
-----
| To:   {firstname} {lastname}      |
| Room: {room}                   |
| _____|
```

We have provided the new Record Definition with dashed lines. We've also provided it with a ".new" Print Design Command--this serves to begin each mailing label on a new "page" (from a computer's eye-view, a label is a page, albeit a short one). Also notice that we have drawn only three fields into the new label, leaving out the fields containing telephone number and age.

Our source file is called "empl.1," and the above layout is the working file. Invoking the following command will now achieve

our intended purpose:

WHAT NEXT? select empl.1 end

The newly SELECTED working file will now look like what appears on the following page.

```
<record>
.new

| To: {firstname} {lastname}
| Room: {room}

<end>
.new

| To: Alexander Gilbert
| Room: X-21

.new

| To: Bob Jones
| Room: M-45

.end
.new

| To: Richard Allison
| Room: X-21
```

It is notable that the SELECT command ignores lines in the disk file that are not part of records.

THE <HERE> SPECIFICATION: Ordinarily, the SELECT command will place newly selected records immediately below the Record Definition in the working file. You may have instances in which you desire to place newly selected records at another location in the working area.

In this case, you can use the <here> option in the working file: Place a line bearing "<here>" only in the working area somewhere after the Record Definition and immediately before the site where newly selected records should be placed.

AUTOMATING A SELECT PROCEDURE: If, say, you were obliged to generate such mailing labels frequently--and with a constantly changing source file--then you might want to establish a permanent DO command to carry out the procedure automatically. You would

accomplish this automation by establishing a file on disk called, say, "labels." It might look like this:

```
select names end find .pagesize print nonstop it
<record>
.new

| To:    {firstname} {lastname}
|
| Room: {room}

<end>
.pagesize 7
..<here>
```

Notice that the commands to perform the task of generating the mailing labels has been made into a DO line at the top of this file. With this file on disk your task becomes very easy to accomplish. You merely have to invoke the following command:

WHAT NEXT? get labels do

You might notice in the format described above that the line containing "<here>" has been placed where it appears so that newly selected records will be placed after the ".pagesize" design command, which sets the vertical height of the labels. The line containing "<here>" is made a comment line (i.e., it begins with two periods in column positions 1 and 2) line so that the "<here>" will not itself print.

REFORMATTING AND THE "IT" OPTION: Recall that a SELECT command may be carried out on the working file by using the "it" option in the command statement. Since the working area will contain only one Record Definition, it is not possible to reformat data into a new layout. It is, however, possible to change the justification of fields.

When the "it" option is used with the SELECT command, all lines in the working area which are not part of a record are left in the working area. Thus, for example, it is possible to SELECT from a list in the working area while keeping headings and the like intact. Note that this is contrary to what happens when the selection is made from a disk file, in which case only valid records are inserted into the working area.

Typically the "it" option is used in conjunction with logical tests as a means to pick certain records out of the working area for inspection.

THE <CONTINUE> OPTION

T/Maker permits you to link or chain together disk-files so that a single SELECT command can process not just one but a series of disk-files. This is accomplished by placing the following line on the bottom line of a data-base file.

<continue> filename

Naturally, you can use this method to link file A to file B, file B to file C, and so on. All you need do is place the appropriate continue specification at the end of each file.

The Record Definitions of all such linked files must be identical.

When the "it" option is used, a "<continue>" specification in the working area is ignored.

The COMPLETE Command

DESCRIPTION

The COMPLETE Command lets you transport data from a file on disk into the working file. Such data are transported en masse. The command requires three words:

WHAT NEXT? complete disk-file-name match-field

What does this tell T/Maker? Let's translate the command as T/Maker hears it:

"complete" tells T/Maker that the working file lacks some information, and T/Maker should go and get that information elsewhere.

"disk-file-name" tells T/Maker that the information the working file lacks can be found on the file named here.

You need not specify the disk drive if the file you want is already the default drive.

"match-field" tells T/Maker the name of a field in the working file that does have information already available in it. Therefore T/Maker can use this field's working-file info as the key for sending the right disk-file information to the corresponding records in the working-file.

The working file and the disk file must have corresponding field names, though they need not have the same structure nor precisely the same number of fields. Nonmatching fields in either file are simply ignored.

If the match-field data are missing on a given record no data will be supplied by COMPLETE to that record.

Corresponding data in the disk file will replace any data that may have been found in the working file. If, for example, a single record in the working file happened to contain complete data in one field but the corresponding case in the disk file were blank on that field, then the working file will become blanked out on that field.

THE <CONTINUE> OPTION

T/Maker permits you to link or chain together disk-files so that a single COMPLETE command can search not just one but a series of disk-files for the needed information. This is accomplished by placing the following line on the bottom line of the disk-file-name referenced in the COMPLETE command.

```
<continue> filename
```

Naturally, you can use this method to link file A to file B, file B to file C, and so on. All you need do is place the appropriate continue specification at the end of each file.

The Record Definitions of all such linked files must be identical.

AN EXAMPLE OF COMPLETE

Below are two files. The working file lacks information that the disk-file has. The working file does contain name information, though, so that these names can be used as the match-field. The following command would fill out the working file's data to produce the result shown.

WHAT NEXT? complete b:address name

THE WORKING AREA

<record>	
Name: {name) Weight: {weight }
<end>	
Name: Smith, Arnold	Weight:
Name: Roberts, Philip	Weight:

THE FILE ON DISK: NAMED "ADDRESS"

```
<record>
Name:      {name
Street:    {street
City/State: {city.state
Work Tel:   {worktel
Home Tel:   {homotel
Weight:    {weight
Height:    {height
<end>
Name:      Smith, Arnold
Street:   123 First Street
City/State: Berkeley, CA 94704
Work Tel:  456-7890
Home Tel:  876-5432
Weight:   165
Height:   68
Name:      Roberts, Philip
Street:   345 Fourth Street
City/State: Danville, CA 93101
Work Tel:  345-3210
Home Tel:  234-0989
Weight:   245
Height:   75
```

THE RESULTING WORKING AREA

```
<record>
Name: {name
<end>
Name: Smith, Arnold
Name: Roberts, Philip
} Weight: {weight  }
Weight: 165
Weight: 245
```

Database Limits:

If you are already familiar with T/Maker, you probably know that it is a flexible program and as such, its limits are also flexible within certain bounds. Each database file is naturally limited to the size of the work file. Database files may be linked together, but each separate file is restricted to the size of the work file.

The number of fields and length of each field is flexible and conforms to a general rule-of-thumb. The number of fields depends on the length of each field and the length of the name for each field. The shorter the field length and the short the name for the field, the greater number of field per record. Associated with each field is a fixed amount of overhead.

The general rule is:

20 (overhead) + length of field name + length of field < 3000

For example:

If we have short field names (2) and short field data lengths (8), we could have $(3000 / (20 + 2 + 8)) = 100$ fields. If however, we used long field names (10) and long data fields (80), we could only have $(3000 / (20 + 10 + 80)) = 27$ fields.

Under CP/M-80, constructing a typical Name and Address database, the size of the work file would allow approximately 400 records. Under MS Dos, the work file is about twice as large and would allow a correspondingly larger database.

For most uses of the database, you will not run into any of the limitations.

DATA-BASE EXAMPLES

The following seven examples show more sophisticated applications you can create once you have acquired a working knowledge of the database structure and commands. These examples are not meant to be used by beginners. The following seven examples are:

automatic letter/mailing label	DB-71
two-across mailing labels	DB-72-73
order summary	DB-74
distributions	DB-75
aligning text within records	DB-76-77
project accounting with error checking	DB-78
sales reporting	DB-79

Easier examples are provided throughout the documentation. The tutorial is an excellent place to start or to use as a reference for creating your own data-bases.

Here are some common applications, and where to find them in the documentation:

address list	DB-5-6
product/price/quantity	DB-12-13, DB-26-27
mailing label	DB-15, DB-61-64
telephone directory	DB-18-20
team statistics	DB-28-30
fancy forms	DB-32
bonus allocation	DB-50
order summarizing	DB-56-57

EXAMPLE 1: The Instant Letter/Mailing Label

A person has a file of addresses called "ADDRESS" with name and address information as well as a greeting and sign-off message for a letter. As he frequently writes letters, he has set up the file below. By replacing "xxx" with a string contained in the name of the person he is writing, he may execute the command below to prepare the necessary text, print design commands, and aligning wedges for both the envelope and the body of the letter. All that is left to do is to type the text of the letter.

Note: The first line in this file is a "Do line," a string of commands. We have put all of the genuine T/Maker commands in upper-case letters to help the new user distinguish these commands from field names, file names, etc. You needn't follow this practice when constructing your own Do lines. (For further explanation of Do lines, consult the File Management section of the T/Maker manual).

```
SELECT address WHEN name CONTAINS xxx END FIND <here> CLIP B EDIT
```

```
<record>
.indent 35
{name
{street
{city }, {state } {zip }
.new
.number 1
.indent 0
.top 2
Page #

.end
{name
{street
{city }, {state } {zip }

(greeting ):

.block
(signoff ),

John Doe
.end
<end>
..<here>
```

EXAMPLE 2: Two-Across Mailing Labels

A person has a file of addresses called "ADDRESS" with the following record definition. Suppose he wants to print labels two across.

```
<record>
Name: {name } }
Line1: {line1 } })
Line2: {line2 } })
Line3: {line3 } })
Line4: {line4 } })
<end>
```

The first step in solving this problem is to develop a new record definition which changes the unit of analysis from one to two records. Instead of changing the record definition in the data base file itself, suppose one creates the file below and calls it "TWO-RECS".

```
<record>
Name: {name } }
Line1: {line1 } })
Line2: {line2 } })
Line3: {line3 } })
Line4: {line4 } })
Name: {xname } })
Line1: {xline1 } })
Line2: {xline2 } })
Line3: {xline3 } })
Line4: {xline4 } })
<end>
<continue> address
```

Selecting from TWO-RECS will be like selecting from ADDRESS except that records will be pulled out two at a time. Notice that it was necessary to think up new names for the extra data in this new double record.

Select always uses the first record definition it finds when selecting from a disk file. The <continue> option is placed in TWO-RECS so the selection will continue with the file ADDRESS even though Select will do this on the basis of the record definition found in TWO-RECS.

If the working area contained the record definition below,
the correct formatting for labels would be achieved.

```
<record>
{name } {xname }
{line1 } {xline1 }
{line2 } {xline2 }
{line3 } {xline3 }
{line4 } {xline4 }
.new
<end>
```

EXAMPLE 3: Order Summary

A data-base on disk named "ORDERS" contains information on the company making an order (company), the date of the order (date), and the amount of the order (amount), as well as a number of other items. Each order is one record. We would like to reorganize some of the data for a report so that we have one record for each company. Within each record we would like to see the date and the amount of each order made by the company.

The command below would do the trick assuming no company had more than ten orders. The trick here is the use of the Group with a second Record Definition to eliminate all but the first "A" record for a series of orders. Try it!

Note: All capitalized words except "A-RECORD" are commands in the following Do line. For your own construction of Do lines, you needn't capitalize commands.

```
SELECT orders END ORDER company FIND A-RECORD GROUP company END 1
```

```
<record>
A{company
B{date <} {amount>} })
B{date1<} {amount1>} )
B{date2<} {amount2>} )
B{date3<} {amount3>} )
B{date4<} {amount4>} )
B{date5<} {amount5>} )
B{date6<} {amount6>} )
B{date7<} {amount7>} )
B{date8<} {amount8>} )
B{date9<} {amount9>} )
<end>

<record> A-RECORD
A{company
<end>
<here>
```

EXAMPLE 4: Using a Blank Example Line to Calculate Distributions

One of the most common uses of the data base commands is to produce distributions of amounts by various categories while showing percentages of the total. In the example below, a table of total Amount by Company is derived. The one curious line here is the blank example line after the end of the table. If this line were left out, the iterative use of Compute (i.e. "compute c") would cause an error on the line that begins with "<record>."

The blank example line causes Compute to ignore all lines until a valid example line is found. Therefore, no error is produced on the second iteration. (Another approach to this problem would have been to put the command "find ex" just before the start of the Compute command to advance the screen beyond the record definition.)

Notice also that by REPLACING the field name "amount" with other field names, different distributions could be generated without changing any of the other instructions.

Note: All capitalized words are commands in the following Do line. For your own construction of Do lines, you needn't capitalize commands.

```
SELECT orders END GROUP company END ORDER D amount COMPUTE C
```

```
<record>
++    (company           ) {amount>} 
<end>
```

Account Breakdown Analysis

```
ex          999,999.99  999.99
uc1          +      fta/%
-----
Company      Amount   Percent
-----
..<here>
-----
jcl          sta
=
Total
-----
avr  Average
-----
ex
```

EXAMPLE 5: Aligning Text Within Records

The data base file on the following page is designed to allow the Align command to be used on it. This means that records with neatly aligned text can be produced. There are some problems with this approach that cannot be completely overcome, but one can learn to tolerate them.

The source of the problem with this Record Definition is that some of the lines in the record have blank line labels. This means that a blank line in the file may be regarded as part of a valid record by the data base commands.

So that the Update command does not create superfluous records when moving around the end of the file, the first line in the record has been made optional. This insures that a blank line (interpreted momentarily as a record) will nonetheless result in only blank lines. If the first line were not made optional, then a blank line (interpreted as a record) would result in a non blank line (i.e. a line beginning with ".1") when placed back into the file by the Update command.

Notice that the first line in the record starts off with a label that makes it a non-aligning line. This ensures that it will be left intact when Align is used.

The fields in the record have been made somewhat bigger than those in the form. This is because one may inadvertently use the Character Insert Keystrokes excessively and push parts of a line too far to the right. Since the record field is longer than the form field, the characters will not be lost as long as the field does not become longer than it appears in the record.

Finally, the line with the Aligning wedges has an * in the first column. Why? If it did not, this line would itself appear to be a line of a record, since it would start with the necessary three blanks. The UPDATE command would use this line as the first record even though the "<here>" option is used, because finding a bonafide record takes precedence over the "<here>" option.

```
<form>
-----
Name: {name} Date: {date}
-----
[ {line1} ] [ ]
[ {line2} ] [ ]
[ {line3} ] [ ]
[ {line4} ] [ ]
[ {line5} ] [ ]
-----
<end>
<record>
.,1 {name} {date} }
{line1} } }
{line2} } }
{line3} } }
{line4} } }
{line5} } }
<end>
* << ->>
<here>
```

EXAMPLE 6: Project Accounting with Error Checking

A person regularly enters times charged to various projects by various staff members. After entering a slew of them, he wants to verify that records have correct staff numbers and project codes.

One way to attack this problem is to build two additional data-bases. One would contain the valid project codes and the other the valid staff numbers. The project file might be called "PROJECTS" and have a record definition like that below.

```
<record>
Project Code: {prjcode} Project Title: {prjtitle} )
<end>
```

The staff file might be called "PEOPLE" and have a record definition as follows.

```
<record>
Staff Number: {staff } Name: {name} )
<end>
```

Finally, suppose the time record had the following rules and record definitions.

```
<rules>
error = ''
error = ERROR when prjtitle = '' or name = '' end
prjtitle = ''
name = ''
<end>
<record>
Staff Number: {staff } Project Code: {prjcode} ) Hours: {hours} )
Error: {<error>} {##name} {##prjtitle} )
<end>
```

After entering the data, the command sequence below would put the string "ERROR" into the error field for each record with either an unknown project title or staff name. Notice that the name and project title fields have been kept to one character, since one really doesn't care what the values of these things are but only whether they are blank or not.

```
COMPLETE projects prjcode COMPLETE people staff SET
```

The Update command could be used to locate the incorrect records by searching for the string "ERROR".

EXAMPLE 7: Quarterly Sales Reporting

A file of orders contains the amount, month, and year of an order as well as the type of customer (A, B, or C). It is desired to show the total amount of orders by quarter and by type of customer using the presentation below. The command below accomplishes the task.

After selecting the orders, the Set command is used to refigure the month as the quarter (remember that Set rounds numbers just like Compute). Other rules are used to position the amount in the correct column of the table based on the type of customer. After this, Group, Order, and Compute are used in a straightforward fashion.

Note: All capitalized words are commands in the following Do line. For your own construction of Do lines, you needn't capitalize commands.

```
SELECT orders END SET GROUP year month END ORDER month ORDER year COMPUTE

<rules>
    month 99 = (month + 1)/3
when type = B
    amount1 = amount
    amount = 0
when type = C
    amount2 = amount
    amount = 0
when type <> ''
    type = ''
<end>

<record>
+      (year){month} (>amount )  (           )  (           )  (<type>
<end>

ex          99,999,999  99,999,999  99,999,999  99,999,999
zv          -          -          -          -
uc1         +          +          +          =
-----+
Year   QTR      Type A      Type B      Type C      Total
-----+
..<here>
-----+
=     Total
-----+
```

SPREADSHEET

T/Maker provides you with the ability to carry out simple or complex mathematical calculations. This is done in a uniquely visual way in T/Maker--it is done by, essentially, making a picture of the calculation you want to perform on the screen and then invoking the Compute command, which carries out the calculation.

It is a very easy-to-use and powerful approach once you get the hang of it.

The Compute command is the heart of T/Maker's calculation system. Two other commands, Clean and Combine, are also of use and described in this section.

COMPUTE

You do calculations in two steps: First, you lay out on the screen a visual representation of the calculation to be performed (this is done while you are "inside the editor"). Second step, you quit the editor and type COMPUTE <RET>.

In a moment your calculation will be carried out.

In general, the key to the use of the Compute command is learning how to lay out calculations properly. Wherever possible T/Maker simply requires that you use ordinary mathematical symbols in their ordinary meaning. Thus, oftentimes you will need to do little more than write the desired calculation as if you were working on a chalkboard. But, as might be expected, T/Maker may also require you to learn some new rules for laying out calculations.

CLEAN

Calculations are laid out by means of equations that you inscribe on the screen. Of course, you will also provide the values that correspond to the variables and constants in the equation. Often, once the calculation you wanted is completed, you will want to remove the calculation equation from the resulting table. This is easily done by using the Clean command (or, alternatively, the ".clean" design command).

COMBINE

A table in the working area and a table on disk can be added together or subjected to some other sort of calculation by means of the Combine command. The results of this calculation become the working area, displacing the working area you started with.

TABLE OF CONTENTS

THE COMPUTE COMMAND	ES-1
Expressing Calculations in T/Maker's Lingo	ES-1
Some Basics	ES-1
T/Makers Order of Calculation	ES-3
T/Maker's Terminator Symbols	ES-5
Summary	ES-7
Vertically Arranged Calculations	ES-8
The Vertical Calculation Strip	ES-8
The Example Line	ES-9
Trying Out Vertical Calculations	ES-10
Labels	ES-12
Multiple Columns	ES-13
Using the Same Position for a New Strip	ES-14
Linking Vertical Calculation Strips	ES-14
Two or More Terminators on One Line	ES-16
Using Three-Letter Terminator Symbols	ES-17
An Example and Exercise	ES-18
The Zero Values Line	ES-19
Horizontally Arranged Calculations	ES-21
The Horizontal Calculation Line	ES-21
Some Basics	ES-22
Zones for Mathematical Symbols	ES-24
More Than One Symbol in a Zone	ES-25
Calculation Line Control Codes	ES-27
Calculation Line Sequence Numbers	ES-35
The Order Of Calculations	ES-36

Restricted Calculations	ES-37
The Constant Calculation Line	ES-38
An Example of the Use of Constants	ES-40
Converting Whole Tables To Percentage Figures	ES-42
Special Notations	ES-44
Cumulate Values	ES-45
Absolute Changes	ES-46
Percentage Changes	ES-46
Moving a Column to the Right	ES-47
Moving a Column to the Left	ES-47
Column Percents	ES-47
Growth Rates	ES-48
Compounded Projections	ES-50
Straight Line Projections	ES-50
Net Present Value	ES-51
Special Operations	ES-52
Rounding	ES-51
Storing and Fetching Values	ES-55
Passing Values	ES-60
Iteration	ES-63
Other Topics	ES-65
Print Design Commands	ES-65
Advanced Example Line	ES-65
Invisible Columns	ES-65
Changing the Example Line	ES-66
Placing Numbers Below Example Lines	ES-67
A Few Words On Vertical Calculation Strips	ES-69
Entering Strips	ES-69
Invisible Rows	ES-70
Debugging Control Codes and Sequence Numbers	ES-70
Errors and Limits	ES-73
Complete Examples	ES-74
 THE CLEAN COMMAND	 ES-93
 THE COMBINE COMMAND	 ES-96

THE COMPUTE COMMAND

The Compute command allows the user to carry out mathematical calculations.

Such calculations are always done in two steps: first, by laying out the calculation problem in a table or visual form (while one is inside the editor), and, second, by quitting the editor and invoking the Compute command.

The key to learning how to use this command is understanding how to lay out calculation problems so that T/Maker can properly understand your intentions.

EXPRESSING CALCULATIONS IN T/MAKER'S LINGO

Some Basics:

Back in grammar school arithmetic problems were written out in the following way:

$$\begin{array}{r} 1 & 2 \\ +1 & \times 2 \\ \hline 2 & 4 \end{array} \quad \text{or} \quad \begin{array}{r} 7 - 2 = 5 \\ 10 / 5 = 2 \end{array}$$

Notice that the two calculations on the left are set up in a vertical direction and the two on the right are set up in a horizontal direction.

It will be useful to have a little, standard vocabulary for the parts of grammar school notation. Let's call the numbers that go into the calculation "values"; let's call the plus, minus, multiplication, and division signs "mathematical operators" or just "operators"; let's call the underline or the = sign that end calculations "terminator symbols"; and, finally, let's call the result "the result."

Now, let's see how T/Maker calculations need to be written out in these terms.

First of all, calculations in T/Maker can be laid out in either the horizontal or the vertical direction (or both, simultaneously).

Secondly, every value in a T/Maker calculation has an associated operator. So, for example, if we consider a simple problem, T/Maker would need to see it expressed not as:

<u>Ordinary Notation</u>	<u>T/Maker Notation</u>
$\frac{1}{+1}$	but instead, + 1 = 2

And if this same problem were written out in the horizontal direction, it would be expressed not as,

<u>Ordinary Notation</u>	<u>T/Maker Notation</u>
$1 + 1 = 2$	but instead, + + = 1 1 2

The key things to grasp are (1) in T/Maker, each value gets its own associated operator.

And (2), the chain of operators in a given calculation is separated off from the associated chain of values. In other words, operators are put in one place and values in another.

If the calculation is laid out vertically, then the chain of operators runs down the left side of the chain of values, and if the calculation is laid out horizontally, then the chain of operators lies above the chain of values.

By the way, there are four legal operators in T/Maker equations:

- + for addition
- for subtraction
- * for multiplication
- / for division

So you see, from the user's vantage point, writing out calculation problems in T/Maker is going to be much the same as writing them out on a piece of paper!

So far, so good. Let's push on.

T/Maker's Order of Calculation:

It is essential that you learn the order in which T/Maker goes about reading the mathematical operators in an equation.

Suppose that we wanted to solve a problem with more than two values in it, like this one:

Add 8 and 6, divide the sum by 7, multiply the result by 42, multiply the result by .5

or, expressed symbolically,

$$8 + 6 / 7 * 42 * .5$$

One of the key features of this problem is that it involves a certain order of calculation. Obviously, it involves some addition, some multiplication, and some division, and operations must be conducted in a proper sequence or the outcome will be wrong.

Now, the ordinary way to go about indicating this order is to supply the equation with parentheses, like this:

$$[(8 + 6)/7] * (42 * .5) = 42$$

All of a sudden, the order becomes quite clear.

T/Maker, as it happens, does not use parentheses. Instead, it follows a simple computational rule. In any calculation formula four steps are followed:

First, all additions are carried out

Second, all subtractions are carried out

Third, all multiplications are carried out

Fourth, all divisions are carried out

At first glance this might seem a mode of procedure bound to produce confusion and incorrect answers! But, in fact, it works quite well, and is easy to use once you get used to it.

Consider for example the problem described above. Setting it out vertically, the problem takes the following appearance:

```
+   8  
+   6  
/   7  
*  42  
* .5  
=
```

Following the order of calculation already described, T/Maker would proceed as follows:

1st, do additions: add 8, add 6, summing to 14

2nd, do subtractions: no subtractions, total remains 14

3rd, do multiplications: multiply 14 by 42 by .5, product is 294.

4th, do divisions: divide 294 by 7, giving 42

A more mathematical statement of this rule is the following:

$$\text{Result} = \frac{[(\text{sum of } +\text{'s}) - (\text{sum of } -\text{'s})] \times (\text{product of } *\text{'s})}{(\text{product of } /\text{'s})}$$

Note that if such a calculation contains no plus or minus signs, the result is certain to be zero.

As we will see, fixing the order of calculation in this way does not pose difficulties for calculation, and, in fact, makes things quite easy to do!

One benefit of the fixed order of calculation is that problems of this type can be arranged in an order that reflects our preferences for presentation. For example, the previous result could have been achieved with the problem stated like this:

```
/ 7  
* .5  
+ 6  
* 42  
+ 8  
=
```

T/Maker's Terminator Symbols:

It was mentioned above that grammar school notation contains two terminator symbols -- the equal sign (in horizontal calculations) or the underline (in vertical calculations) -- which terminate a given calculation and produce the result.

It was also mentioned that T/Maker uses only four mathematical operator symbols (+, -, *, /).

Now, this might at first glance seem like a relatively small selection of mathematical operators. How, for example, shall we take a square root or find the cosine of an angle? Well, fear not! T/Maker handles these sorts of calculations in the terminator symbol of a calculation.

Unlike grammar school notation, T/Maker notation provides a great variety of terminator symbols (19 of them). You can put a simple equal sign (=) at the end of a calculation, which will give you the simple result. Or, you can put in the same place one of 18 other terminator symbols, each providing a special result. Think of these other terminator symbols as performing "one last calculation" on the ordinary result.

REGULAR TERMINATOR SYMBOLS

<u>Symbol</u>	<u>Modification</u>
=	the simple result
%	the result is multiplied by 100
#	the result is divided by 100
sfh	the result is subtracted from 100
sfo	the result is subtracted from 1
rec	the reciprocal of the result is taken
sqr	the square root of the result is taken
abs	the absolute value of the result is taken
exp	e to the power of the result is taken
log	the natural log of the result is taken
l10	the log to the base 10 of the result is taken
sin	the sine of the result is taken (radians)
cos	the cosine of the result is taken (radians)
tan	the tangent of the result is taken (radians)
atn	the arctangent of the result is taken (radians)
ply	the maximum of zero and the result (positives only) is taken
nly	the maximum of the negative of the result and zero (negatives only) is taken
sgn	the sign of the result becomes the result (+1 for positive numbers, -1 for negative numbers, zero for zero).
avr	the result is divided by the number of plus signs in the equation (if the equation only has plus signs, the result would be the average).

The results of using three of these terminator symbols to end a general calculation are shown below.

+ 4	+ 8	/ 20
/ 8	+ 14	+ 5
% 50	+ 2	+ -3
	avr 8	rec 10

So far, we've looked at mathematical operators and "regular" terminator symbols. Perhaps you guessed that the existence of "regular" terminators implies the existence of "irregular" or "special" terminators as well. You were right!

T/Maker provides a set of five "Special Terminator Symbols". These are like the Regular Terminators in that they generate a result, bringing a given calculation to a close. But they are unlike Regular Terminators in that they ignore the conventional meanings of the mathematical operators in the equations they terminate. Instead, the operators are used merely to flag or to designate values the Special Terminator Symbol will consider.

SPECIAL TERMINATOR SYMBOLS

- + 5 The "max" terminator selects the maximum of the numbers associated with the "+" and the "*".
* 9
max 9

 - + 5 The "min" terminator selects the minimum of the numbers associated with the "+" and the "*".
* 9
min 5

 - + 10 The "~~" terminator produces the percent change in going from the value associated with the "+" to the value associated with the "*".
* 12
~ 20

 - + 2 The "^^" terminator produces the value identified by the "+" raised to the power identified by the "*".
* 3
^ 8

 - + 9 The "cnt" (for "count") produces a count of the number of plus signs in the calculation.
+ 3
+ 8
cnt 3
-

Summary:

Let's review for a moment what we've covered. First, T/Maker's approach to calculation uses ordinary symbols, but separates chains of mathematical operators from their associated values. Next, we learned that T/Maker always follows a fixed order of calculation when it proceeds through a formula: additions first, subtractions second, multiplications third, divisions fourth. Next, we learned that calculations can be terminated by 19 different kinds of Regular Terminator Symbols or 5 different kinds of Special Terminator Symbols.

Now! On to vertically-arranged calculations!

VERTICALLY ARRANGED CALCULATIONS

The Vertical Calculation Strip:

As we've seen already, vertically arranged calculations involve two kinds of columns: a column of mathematical symbols (made up of mathematical operators and a terminator symbol) and a column of values. Like this:

<u>Symbols</u>	<u>Values</u>
+	12
+	5
-	7
=	

Let's call the column of symbols a "Vertical Calculation Strip" or "Strip" for short.

There are a couple of rules governing the use of Vertical Calculation Strips:

RULE 1: All the elements of a single Strip must fall in a single vertical position.

So, for instance, Example A's Strip (below) is correct and Example B's Strip is incorrect (in B, the plus sign associated with the 5 is out of line):

Example A

+ 12
+ 5
- 7
=

Example B

+ 12
+ 5
- 7
=

RULE 2: All Vertical Calculation Strips must be placed in one of the first seven horizontal positions.

Relatedly, columns of values must be placed in positions 8 or higher.

You should perhaps think of the first seven positions of all lines as zone reserved Strips.

Positions Reserved for Strips	Positions Reserved for Values (8 and higher . . .)
-------------------------------------	--

1234567 8 and higher

The Example Line*:

We come, now, to the final ingredient necessary to carry out a vertically arranged calculation, the "example line."

The example line is a horizontal line consisting of "example" column positions one through seven, followed by one or more groups of 9's called "model numbers." For example, an example line might look like this:

example 999

An example line is always at the top of a T/Maker calculation. It tells T/Maker that a calculation problem is beginning. It also has two other purposes:

First, it shows T/Maker where to look for the numerical values in the calculation. It is the nines that define the vertical zones in which T/Maker will look for values.

* The "Advanced Example Line" subsection describes other capabilities related to the Example Line.

example	999
	000
	000
	000
	000
	000
	000
	000
	000
	000

The zone indicated by the nines stretches on indefinitely down the file--only when another example line is encountered is the zone superceded.

Second, it shows T/Maker how the numbers in a given calculation should be presented. For example, if the nines look like this, "999", the values appearing below them will all be conformed to that manner of presentation--namely, they will be presented as whole numbers up to three digits long; if the nines look like this though, "999.99", then values appearing below will be presented carried out to the hundredth's accuracy.

Because these nines serve the function of providing models of how values should be presented, they are called "Model Numbers."

Incidentally, model numbers must begin no sooner than the 8th position.

To repeat, then, an "example line" consists of a line with "example" (which may be abbreviated to "ex" only), followed by one or more "model numbers."

Trying Out Vertical Calculations:

This is a good moment to try out some vertically arranged calculations, both to consolidate what you've just learned and also to learn some new things.

Let's start with a simple problem. Bob has \$875.34 in his checking account, his paycheck for this month is about to be deposited too--it is \$1,419, and he owes \$1,256 in debts this month. How much will he have left over when his debts are paid? Of course, it is a very easy problem. Let's set it up in T/Maker:

First, let's put down the basic facts. We want to add two numbers together and subtract a third. We want to generate a result which is the net amount remaining. So, we begin by placing a Vertical Calculation Strip in, say, position 3, and by placing the associated values* in positions 8 and higher:

1st STEP

```
+      875.34  
+    1,419.00  
-    1,256.00  
=
```

Next, we have to put in an example line to show T/Maker where the values appear, and, also, how they should be presented. Since we want the result printed out to "penny" accuracy we'll use a model number like this, "9,999.99".

2nd STEP

example 9,999.99

```
+      875.34  
+    1,419.00  
-    1,256.00  
=
```

Now, we're ready to go! All we need do is place the cursor at or above the example line, quit the editor, and type in "compute <RET>" in response to the "WHAT NEXT?" prompt.

WHAT NEXT? COMPUTE

* When entering numbers, it is not necessary to justify them or to follow the dictates of the model number with respect to commas or a decimal point. Compute will align the numbers itself. However, the number must be placed entirely in the zone defined by the model number. (See "Placing Numbers Under The Example Line" for some rules and suggestions on how to enter numbers.)

The calculation will now reappear with the result calculated:

RESULT

example 9,999.99

+	875.34
+	1,419.00
-	1,256.00
=	1,038.34

What could be easier!

Let's now introduce some new possibilities:

a. Labels

Suppose that Bob wanted to have his table of expenses include some explanatory labels.

He could, for example, write in the following information:

BOB'S BUDGET FOR OCTOBER

example 9,999.99

+	875.34	CHECKING BALANCE
+	1,419.00	PAYCHECK
-	1,256.00	OUTSTANDING BILLS
=		NET BALANCE

These labels, as it turns out, are perfectly all right. As long as labels do not fall in the first seven positions (reserved for Strips) or beneath the zone of model numbers on rows of data, they are legitimate.

In fact, one might also place the labels as below:

BOB'S BUDGET FOR OCTOBER

example	9,999.99
+ CHECKING BALANCE	875.34
+ PAYCHECK	1,419.00
- OUTSTANDING BILLS	1,256.00
= NET BALANCE	

Notice that labels, once again, have been arranged so they neither fall in the first seven positions nor beneath the zone of model numbers.

b. Multiple Columns

Suppose Bob wants to get some idea of October's budget compared to September's and August's budgets.

T/Maker's Vertical Calculation Strips will happily carry out the same calculation simultaneously for more than one column of figures. In fact, the limit on numbers of columns is twenty-five--in other words, Bob could compare twenty-five months of budgets if he likes.

The calculation now should be laid out as below. Invoking the Compute command carries out all three calculations simultaneously and independently.

BOB'S BUDGET FOR OCTOBER, SEPTEMBER, AND AUGUST

	<u>October</u>	<u>September</u>	<u>August</u>
example	9,999.99	9,999.99	9,999.99
+ Checking Balance	875.34	544.99	1,854.08
+ Paycheck	1,419.00	1,419.00	1,419.00
- Outstanding Bills	1,256.00	1,134.86	1,198.79
= NET BALANCE	1,038.34	829.13	2,074.29

Notice that the example line now includes three model numbers--one for each of the columns of numbers.

c. Using the Same Position for a New Strip

You might think of T/Maker's calculating ability as a kind of shovel device that starts at the top of a vertical calculation and moves downward, collecting new values and operators, until it reaches a terminator symbol which instructs it to dump out whatever value is presently in the shovel. As soon as a terminator is encountered, the shovel is emptied and ready for use again.

Hence, the position used for a Vertical Calculation Strip that has come to an end can be used to start up a new Strip. For example, suppose that Grace wanted to know both how much money she had and how much money she needed to buy a few things she wanted:

GRACE'S PLANNED EXPENSES

example	999.99
---------	--------

MONEY AVAILABLE

+	Piggybank	27.87
+	Wallet	4.05
+	Savings Account	38.67
=	Total Assets	70.59

PRICES

+	Candy bar	0.25
+	Phonograph Record	1.98
+	School Supplies	11.99
=	Total Expenses	14.22

Now, notice that the Strip totalling Grace's available monies is in the same column position as the Strip totalling Grace's planned expenses. Invoking the Compute command (with the cursor at or above the example line) will still carry out both computations.

d. Linking Vertical Calculation Strips

Now suppose that Grace had wanted to calculate how much money she would have left after buying the items she wanted.

This can be very easily done in T/Maker by starting a new Strip in a new position. All we need do is subtract Grace's "Total Expenses" figure from her "Total Assets" figure. This is done as follows:

GRACE'S PLANNED EXPENSES

example	999.99
---------	--------

MONEY AVAILABLE

+ Piggybank	27.87
+ Wallet	4.05
+ Savings Account	38.67
= + Total Assets	70.59

PRICES

+ Candy bar	0.25
+ Phonograph Record	1.98
+ School Supplies	11.99
= - Total Expenses	14.22
= BALANCE	56.37

As the example above shows, T/Maker allows you to "pick up" a result from one Strip and use it as a value in another Strip.

One can also simply set up new Strips to do new computations. For example, suppose that Grace spent the money for the candy bar and the phonograph record from her wallet funds, and desired to know how much, now, remained in her wallet. This could be easily added as an independent Strip. Below, this new little calculation has been made a Strip in position 1.

GRACE'S PLANNED EXPENSES

example		999.99
MONEY AVAILABLE		
+ Piggybank	27.87	
+ + Wallet	4.05	
+ Savings Account	38.67	
= + Total Assets	70.59	
PRICES		
- + Candy bar	0.25	
- + Phonograph Record	1.98	
+ School Supplies	11.99	
= - Total Expenses	14.22	
= Left in Wallet	1.82	
= BALANCE	56.37	

As you grow more proficient with T/Maker's computing capabilities you may find that linking Strips provides a powerful method for carrying out complex computations quite easily.

e. Two or More Terminators on One Line

Take a look once more at the previous calculation, concerning Grace's expenses. Notice, in particular, that there are four terminator symbols (all equal signs), and that no two terminator symbols appear on the same horizontal line. This makes sense. After all, a terminator symbol generates a result which must be placed into the table, and so one would think that only one terminator symbol should appear on a single line.

In general, that is a good rule. But there are instances in which it is handy to have two terminators on a single line. Consider:

When two or more terminators do appear on the same line T/Maker adds the results of the two together, producing the summed results. This affords the user a convenient way to do such sums when they are desired.

For example, suppose that we wanted to compute the total price of something that happened to have a 6% sales tax associated with it. We might proceed as follows:

example	999.99
+ + Price before taxes	100.00
* Sales Tax (@ 6%)	0.06
= = Total Price	106.00

Let's interpret the example just above.

The Strip in position 1 first enters 100 into the T/Maker calculator. Then it multiplies this 100 by 0.06. Finally, the product of this multiplication (6.00), is "dumped out" on the third line, where the terminator symbol appears.

The Strip in position 3 enters 100 into a second T/Maker calculator. It does nothing more than dump out this result (100.00) on the third line, where this Strip's terminator appears.

The results, therefore, consists of both results combined: 6.00 added to 100.00 giving 106.00.

f. Using Three-Letter Terminator Symbols

You may recall that most of the Regular Terminator Symbols and two of the Special Terminator Symbols are three letters long. How, then, can they be placed in a single Vertical Calculation Strip?

Answer: Site the first of the three letters in the Strip column-position. T/Maker will understand.

AN EXAMPLE AND EXERCISE

Now, let's try to understand a real-life application of Vertical Calculation Strips. John Doe recently went to Germany, Switzerland, and Holland on business, and he has prepared his expense account. We won't tell you the kinds of guidelines that govern foreign expenses in John Doe's company. See if you can figure out what he has done and why!

For your convenience, the figures that T/Maker calculates are presented in bold. You might try scanning the table from top to bottom. When you see a row that is associated with a terminator symbol, look back up the strip for the mathematical operators involved. Enter these operators into your thought process in the order they are used by Compute's calculator (additions first, subtractions second, etc.). And when you understand what's going on in that row, move on to the next.

John Doe's Expenses			
	German Marks	Swiss Francs	Dutch Gilders
ex	99,999.99	99,999.99	99,999.99
+	548.50	800.00	335.50
+	245.40	287.60	120.20
+	130.00	450.00	143.00
+	210.50	240.10	120.50
= +	1,134.40	1,777.70	719.20
/	2.02	1.57	1.89
* = /	561.58	1,132.29	380.53
+	3.00	4.00	2.00
Maximum Reimbursement			
* Per Day (\$)	190.00	225.00	200.00
Total Maximum			
* = Reimbursement	570.00	900.00	400.00
Amount To Be			
min + Reimbursed	561.58	900.00	380.53
% Percent Reimbursed	100.00	79.48	100.00

Note: "Min" is a special terminator, so the addition and multiplication signs above it in the strip are used to flag rows and do not imply the usual mathematical operations.

AN ASIDE: THE ZERO VALUES LINE

Before we start a discussion of horizontally-arranged calculations let us briefly mention something called the "Zero Values Line." This is an optional line one may add to a table just below the example line. The Zero Values Line allows the user to specify how values falling on exactly zero should be presented in a table.

When no Zero Values Line is included in a table, exactly-zero values are presented in accordance with the model number for that column. But when a Zero Values Line is provided, exact-zeroes are presented in the manner prescribed by the Zero Values Line. This line begins with a "zv" in positions 1 and 2, followed by the precise character strings desired for exact-zero values placed beneath the appropriate model numbers.

Below, two examples are presented. In Example 1, a zero values line is not used so exact-zero results are reported according to the appropriate model numbers:

EXAMPLE 1

	Monday	Thursday
ex	99,999.99	99,999.99
+	Bread	1.33
+	Milk	3.49
+	Cream	1.95
+	Orange Juice	0.00
+	Coffee	0.00
+	Jam	0.00
+	Meat	14.78
=	Total	21.55
		19.26

In this table, zero means "not purchased," and it might be better presented by a dash or blanks. In Example 2, a Zero Values Line has been added. This Line specifies that exact-zero values in the first column are to be presented as a dash and exact-zero values for the second column are to be presented as blanks.

EXAMPLE 2

	Monday	Thursday
ex	99,999.99	99,999.99
zv	-	
+	Bread	1.33
+	Milk	3.49
+	Cream	1.95
+	Orange Juice	-
+	Coffee	12.23
+	Jam	1.23
+	Meat	14.78
=	Total	21.55
		19.26

Any characters may be used as a zero value: "none," "ZIP!," etc.

Incidentally, values falling close to, but not exactly upon, zero are always presented in the fashion prescribed by their model numbers, even if the result is so small that all of the digits reported in accordance with the model number are zeroes.

Tip: "Rounding," described later, can be used to force numbers which are almost zero to become exactly zero, so that the zero values specifications will be used to present them.

HORIZONTALLY ARRANGED CALCULATIONS

The Horizontal Calculation Line:

The "Horizontal Calculation Line" is the horizontal equivalent of the Vertical Calculation Strip. Happily, many of the same ideas that you just learned about Strips will apply equally well to Horizontal Calculation Lines.*

It happens, though, that Horizontal Calculation Lines (or Lines, for short) are a bit more powerful than Strips--which is to say, you can do more with them. To characterize the extra strength Lines have, it is useful to introduce two new definitions -- "General Calculations" and "Restricted Calculations."

In the last section you learned about T/Maker's mathematical operators (four of them) and Regular and Special Terminator Symbols. Now, here comes the first definition: Any equation made up of operators and Regular or Special terminators is called a "General Calculation." It is called "General" because it can be generally applied--you can use it in vertically-arranged or horizontally-arranged or even cross-table calculations (see the Combine command).

"Restricted Calculations," on the other hand, are special calculating powers that can be used only on Horizontal Calculation Lines. That is why they are called "restricted." In a little while we will describe Restricted Calculations. For the moment, though, let's consider some of the basic facts about horizontally arranged calculations.

* Specifically, in both Strips and Horizontal Calculation Lines, (1) the order of calculation is the same (all additions first, then all subtractions, all multiplications, and finally, all divisions); (2) the same operators and terminator symbols are used; (3) once a Strip or Line is ended by a terminator symbol, a new Strip or Line can be started up; and, (4) like Strips, Calculation Lines can be linked together.

a. Some Basics

Let's consider our old friend, $1 + 1 = 2$. We learned earlier that T/Maker always associates a mathematical symbol with each value in a calculation. We also learned that when calculations are laid out horizontally, the mathematical symbols appear above the values. So, laying out our problem, it might look like this:

$$\begin{array}{ccc} + & + & = \\ 1 & 1 & 2 \end{array}$$

Just as in Strips, we are going to need an example line, too, to show T/Maker where values are going to appear.

ex 99 99 99
 + + =
 1 1 2

You should notice, by the way, that in horizontally arranged calculations the example line not only shows where values appear, but also where the mathematical symbols will appear, too. So, for example, the arrangement below, which leaves the model numbers in the same place but moves the operators to the end of the Calculation Line, would not accomplish the calculation we want:

ex 99 99 99
 ++=
 1 1 0

This turns out to be an important point, which will be discussed more later on. But, let's get back to the main line.

Based on our experience with Vertical Calculation Strips one might well assume that we are now ready to invoke Compute and see our calculation carried out. But, in fact, two more things need to be added to the lay-out of the calculation before we can proceed.

The first is simple: We need to add a plus sign (or some other symbol) in one of the first seven positions of the row of values. This shows T/Maker that values for calculation will appear on this row:

ex	99	99	99
	+	+	=
+	1	1	2

The second thing needed is a little more difficult to understand, but not too hard! It is really a combination of two things called a Calculation Line Control Code and a Calculation Line Sequence Number. The Control Code ("ac") appears in the first two positions of the Calculation Line and the Sequence Number ("1") follows immediately after it:

ex	99	99	99
aci	+	+	=
+	1	1	2

The Control Code and Sequence Number give Calculation Lines two powers that strips don't have. First, a Calculation Line can be selectively applied (remember a strip is always applied to all columns in the table); and, a sequence among Calculation Lines can be specified (a strip that terminates first is always calculated first). We will have to learn more about Control Codes and Sequence Numbers in a little while.

For the moment the important point to grasp is that we now have the form that all horizontal calculations will take:

ex	99	99	99
aci	+	+	=
+	1	1	2

It remains to explore in a little more detail some of the special features of horizontally-arranged calculations: the zone-defining behavior of the Example Line, Calculation Control Codes and Sequence Numbers, and the mysteries of "Restricted Calculations."

b. Zones for Mathematical Operators and Terminator Symbols

You'll remember about Vertical Calculation Strips that the mathematical operator or terminator symbol associated with a particular value fell on precisely the same row that the value fell upon. We wouldn't have written a problem out like the one below, because one of the 1's isn't on the same row as its associated plus sign.

ex	99
+	1
+	
=	1

When you do Horizontal Calculations, though, you gain a bit more flexibility. For example, suppose we want to add together 10,090, 89,765, and 45,998. We'll set up the problem like this:

ex	999,999	999,999	999,999	999,999
aci	+ + + =			
+	10,090	89,765	45,998	

The above is a perfectly adequate way to set up the problem for T/Maker. What is worth noting at this point, though, is that the mathematical symbols might have been set up at any spots beneath their associated model numbers. So, for example, it would have been just as good to place them like this:

ex	999,999	999,999	999,999	999,999
aci	+ + + =			
+	10,090	89,765	45,998	

The result would still be the same. In fact, the model numbers in horizontal calculations serve to establish a wide zone in which mathematical symbols can legitimately appear.

ex <-----999, 999<----999, 999<---999, 999<---999, 999---->
aci + + + =
aci ++ +=

These zones extend the zone for placing numbers to the left until the next model number is encountered, or, in the case of the first column, to position 8. The last column zone is also extended to the right. The two Calculation Lines above describe the same calculation!

Now, naturally, you will usually want to arrange your mathematical symbols clearly and unambiguously beneath model numbers.

Nevertheless, this little feature of horizontal calculations is important to know, and can come in handy!

c. More Than One Mathematical Symbol Can Be Placed in A Value's Zone

Unlike Strips, Horizontal Calculation Lines permit you to place more than one mathematical symbol in the zone of a single value. Maybe this is easiest to grasp by using a simple example.

Suppose that we wanted to calculate the formula $(B-A)$ divided by B. Let's set it up:

	A	B	$(B-A)/B$
ex	999.99	999.99	999.99
aci	-	*/	=
+	25	75	.67

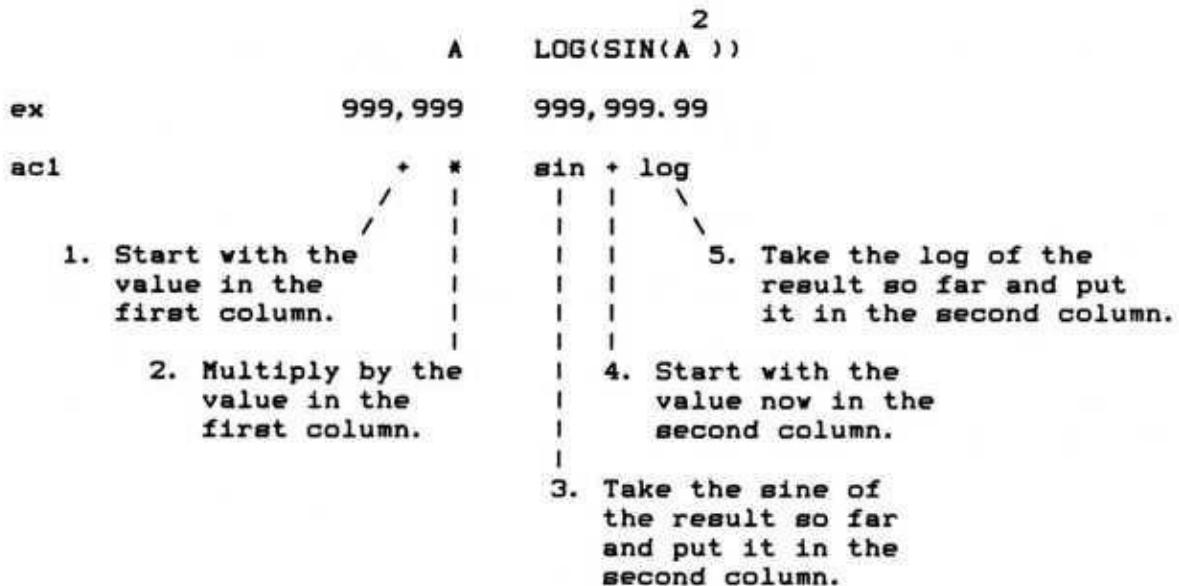
Now, let's interpret it. We have set the problem up in three columns of information: A's value in the first, B's value in the second, and the result of the calculation in the third. How does T/Maker read this Calculation Line, then?

Well, it reads the Calculation Line like this: First, it begins with nothing, then it looks for additions. It finds a plus sign under B and puts the value associated with that sign (75) into its calculator. Next, it looks for subtractions--and

finds the 25 under A. This 25 is subtracted from 75, leaving 50 in the calculator. Next, it looks for multiplications and finds none. Next, it looks for divisions. It finds the 75 under B, and carries out the requested division: the 50 in the calculator divided by the 75 in the B-column gives a result of .67. Finding no more operators, T/Maker moves to the terminator symbol, dumping out the result in the C column.

With a little experience, the capacity to put more than one mathematical symbol in the same value's zone can prove to be very handy and quite powerful.

One more point: Nothing stops you from putting terminator symbols in with mathematical operators under the same value zone. For example, below, two regular terminators are used on a single Calculation Line: "sin" and "log". The second General Calculation uses the value computed by the first General Calculation and ultimately replaces it with its own result. Perhaps, this Calculation Line is worth a diagram.



Note: While not to be encouraged, General Calculations can be split across Calculation Lines. In the example below, the first column is set equal to the sum of the second and third columns. When this approach is used, care must be taken that both Calculation Lines use the same Control Code (so they will apply to the same rows) and that consecutive Sequence Numbers are used (so that no calculations are done in between the two lines). Store and Fetch, described later, can be used to accomplish the same effect in a more disciplined fashion.

	A + B	A	B
ex	99999	99999	99999
ac1		+	+
ac2	=		

CALCULATION LINE CONTROL CODES

A Horizontal Calculation Line consists of a Calculation Control Code in the first two positions, a Sequence Number starting in position three, and the mathematical symbols of the calculation.

ac1 + + =

The Calculation Control Code is the "ac" in the above Calculation Line. (In the next section, the meaning of the number will be described).

The Control Code specifies which rows of values the Calculation Line equation will apply to. A Calculation Line can only apply to rows beneath it. There are five options for Control Codes.

<u>code</u>	<u>equation will apply to . . .</u>
ac	all rows of values
uc	all rows except those derived from other rows by means of a Calculation Strip
rc	only rows derived from other rows by means of a Calculation Strip
jc	only the next row of values below the Calculation Line
sc	this code suspends computations for the next row only
cc	this code suspends computations for all following rows

The power to specify to which lines a Calculation Line will apply provides a considerable increase in computational ability.

Don't feel too bad, though, if at first the utility or the operation of these codes does not seem entirely obvious. They will take a little getting used to!

In order to get your feet a bit wet, though, it is a good idea to consider a few examples of these codes' uses. As you grow more familiar with T/Maker, you will probably find yourself using Control Codes more and more! For now, though, the basic objective is to make their acquaintance.

* These code letters stand for the following names:

ac	"always compute"
uc	"usually compute"
rc	"rarely compute"
jc	"just compute" [the next row only]
sc	"suspend compute" [for the next row only]
cc	"cancel compute"

EXAMPLES OF CALCULATION CONTROL CODES

The example below consists of a simple set of multiplication problems: 1 times 1, 2 times 2, and 3 times 3. This is expressed in the Horizontal Calculation Line as the equation, "+ * =".

A Strip along the left side has also been employed, summing the values in the table's columns.

ex	99	99	999
zv			
??1	+	*	=
+	1	1	
+	2	2	
+	3	3	
=			

This, of course, is a very simple arithmetic problem. But it is worth careful attention, because it contains several important features.

First of all, you should notice that simple as it is, the table contains a fundamental dilemma. This dilemma can be made clearer to the eye by carrying out all but one of the table's calculations:

ex	99	99	999
zv			
?1	+	*	=
+	1	1	1
+	2	2	4
+	3	3	9
=	6	6	?

Notice that we have carried out the three multiplications called for by the Horizontal Calculation Line, placing their results, in bold, in the third column. And, notice that we have carried out the addition called for by the Strip, placing two results in the fourth row of the table.

The dilemma arises in that one remaining cell, with the questionmark in it. What value should be put in that cell? If we follow dictates of the Calculation Line, then we should multiply 6 times 6 giving the result of 36. If, on the other hand, we follow the dictates of the Strip, we should add together 1, 4, and 9, producing the result of 14.

Now, as it happens, our problem is not so much trying to figure out which result is the right one--there will be situations in which one is right and situations in which the other is right. No, our problem is that of pointing out to T/Maker which thing we want it to do.

Here's where Calculation Control Codes come in very handy!

Using "AC" ("Always Compute")

"AC" tells T/Maker to carry out the equation given on the Horizontal Calculation Line on ALL ROWS of a table. Thus, if we put this code into our table it would come out like this:

ex	99	99	999
zv			
AC1	+	*	=
+	1	1	1
+	2	2	4
+	3	3	9
=	6	6	36

Using "UC" ("Usually Compute")

"UC" tells T/Maker to carry out the equation given on the Horizontal Calculation Line on ALL ROWS EXCEPT ROWS DERIVED BY A VERTICAL CALCULATION STRIP. Another way to say the same thing is carry out the calculation on all rows except those with a terminator symbol in one of its Strips. In our problem, though, the "uc" instruction code would tell T/Maker to carry out the HCL's multiplications on the first, second, and third rows, but not the fourth. There, on the fourth, the Strip would take precedence. Thus, the "uc" code would produce the following result:

ex	99	99	999
zv			
UC1	+	*	=
+	1	1	1
+	2	2	4
+	3	3	9
=	6	6	14

Using "RC" ("Rarely Compute")

"RC" tells T/Maker to carry out the equation given on the Horizontal Calculation Line ONLY ON ROWS THAT ARE DERIVED BY A CALCULATION STRIP. "RC", then, is a bit like the complement of "uc". Another way to say the same thing is only rows with a terminator symbol in one of its Strips. In our problem the "rc" instruction code would tell T/Maker to carry out the HCL's multiplications on the fourth row only. Thus, the "rc" code would produce the following result:

ex	99	99	999
zv			
RC1	+	*	=
+	1	1	
+	2	2	
+	3	3	
=	6	6	36

Using "JC" ("Just Compute" Once)

"JC" tells T/Maker to carry out the equation given on THE NEXT ROW OF VALUES ONLY. In our problem, then, the "jc" instruction code tells T/Maker to carry out the HCL's multiplication on the first row only. Thus, the "jc" code would produce the following result:

ex	99	99	999
zv			
JC1	+	*	=
+	1	1	1
+	2	2	
+	3	3	
=	6	6	1

(Notice that the Vertical Calculation Strip will have only a "1" in the third column to put into its total for that column--thus, that becomes the Strip's total for the third column.)

Ordinarily, the "jc" code is used in order to interject one computation, for a specific row of values, into the sequence of calculations in a table. So, for example, in the following table an "ac" or "always compute" HCL tells T/Maker to divide the number in the first column by the number in the second, dumping out the result into the third column. Because "ac" was used, this calculation will be carried out in all of the tables rows of values.

But, as you can see, a "jc" HCL has been introduced just above the row that would have divided 2 by 2. This calculation calls for the values to be multiplied by each other. And so, the result of 4 is produced on that row of the table.

Notice, finally, that the next row of table returns to the control of the "ac" Control Code: 3, once more, is divided by 3 in accordance with the "always compute" Control Code. So, too, is the line containing two 6's.

ex	99	99	999
zv			
ac1	+	/	=
+	1	1	1
jc2	+	*	=
+	2	2	4
+	3	3	1
=	6	6	1

Using "SC" ("Suspend Compute")

"SC" tells T/Maker to suspend computation for the next row of the table. So, Horizontal Calculation Lines beginning with an "sc" code do not specify a calculation at all, but merely instruct T/Maker to skip over the next row and continue calculations on the row following it:

ex	99	99	999
zv			
ac1	+	*	=
+	1	1	1
SC1	+	2	2
+	3	3	9
=	6	6	36

Using "CC" ("Cancel Compute")

"CC" tells T/Maker to cancel computation for all following rows in the table. Like the "sc" code, "cc" codes do not include a calculation formula but merely tell T/Maker to stop calculation for all following rows. In our exemplary problem a "cc" would have the following effect:

ex	99	99	999
zv			
ac1	+	*	=
+	1	1	1
CC1			
+	2	2	
+	3	3	
=	6	6	1

Notice, by the way, that the Vertical Calculation Strip's calculation is still carried out. Of course, the third column is comprised of only the "1" generated by the first row of the table. Thus, the correct total for the third column is "1" as well.

Incidentally, if either "sc" or "cc" is used without a Sequence Number following it then all Calculation Lines above it (no matter what their Sequence Number) will not be applied to the next row ("sc") or all following rows ("cc").

Some Advice For Beginners:

There is no right or wrong Control Code for a particular problem. Every problem can, in fact, be done only using the "jc" code and putting each appropriate Calculation Line ahead of each row it applies to. The other codes are just a convenience to simplify the presentation of a problem.

When beginning with T/Maker, one might start with the ac, jc, sc, and cc codes, since these are defined in terms of "all rows below" or "only the next row." This sort of spacial relationship is easy to master. The use of the uc and rc codes requires not only spacial considerations (i.e., the rows below) but also a consideration of the particulars of the rows themselves (data or derived). Thus, they are a little harder to master.*

CALCULATION LINE SEQUENCE NUMBERS

The number following the two-letter Calculation Control Code is called a Sequence Number. It is always placed immediately after the Control Code. This number determines the order in which Calculation Lines are applied to rows of values when two or more Calculation Lines are used. Even when only one Calculation Line is used, though, it is necessary to include a number with the Calculation Control Code.

The sequence number may range from 0 to 20 (normally, zero is not used--it is included for use on occasions when you want to add a calculation in before a series already created, so that you don't have to change all the sequence numbers already in place). If the sequence number of a calculation line uses the same number as used by a previous calculation line, then the new calculation line replaces the previous one.

In the table below, the calculation lines have purposely been presented in reverse order to highlight the fact that it is the Sequence Number and not the order of appearance that determines the order in which calculation lines are applied to a given row.

	Yes	No	Total	Yes	No	Total	Grand Total
px	999	999	999	999	999	999	999
sc2			+			+	=
sc1	+	+	=	+	+	=	
	1	2		3	4		
	3	2		1	5		

* See "Debugging Control Codes and Sequence Numbers" for a way to let your table, itself, tell you which Calculation Lines will be applied to which rows.

THE ORDER OF CALCULATIONS

As we have already seen, there is an interplay between Calculation Strips and Calculation Lines. The following rule describes the relative order in which Strips and Lines are calculated.

Tables are calculated one row at a time moving from the top to the bottom of the table. For each row, the order of calculation is the following:

- 1) If the row is derived from other rows, the appropriate Calculation Strip is used to derive all columns of the row.
- 2) The appropriate Calculation Lines (as determined by Control Codes) are then applied in order (as determined by Sequence Numbers).

AN EXAMPLE

Calculation Lines Applied

		1	2	3	4
AC2	Calculation Line 2 (applies to all rows)				
UC3	Calculation Line 3 (applies to data rows only)				
RC4	Calculation Line 4 (applies to derived rows only)				
+	Row A (Data)	✓	✓	✓
+	Row B (Data)	✓	✓	✓
= +	Row C (Derived)	✓	✓	✓
JC1	Calculation Line 1 (applies to next row only)				
*	Row D (Data)	✓	✓	✓
=	Row E (Derived)	✓	✓	✓

The series of calculations for the table is:

1. Row A is processed by applying in order Calculation Lines 2 and 3.
2. Row B is processed by applying in order Calculation Lines 2 and 3.
- 3a. Row C is derived from rows A and B according to the Calculation Strip.
- 3b. Row C is processed by applying in order Calculation Lines 2 and 4.
4. Row D is processed by applying in order Calculation Lines 1, 2, and 3.
- 5a. Row E is derived from rows C and D according to the Calculation Strip.
- 5b. Row E is processed by applying in order calculation Lines 2 and 4.

RESTRICTED CALCULATIONS

We saw earlier that T/Maker provides two broad kinds of calculations, General Calculations (made up of operators and regular or special terminators) and Restricted Calculations. We also learned that General Calculations can be used vertically, horizontally, or even across tables (see Combine command), whereas Restricted Calculations can be used only on Horizontal Calculation Lines.

This section is about Restricted Calculations.

Restricted Calculations provide methods for accomplishing a number of things that are difficult to do with only the tools provided by General Calculations. Three sorts of capabilities are provided:

- (1) you can easily place constants into calculations,
- (2) you can use a number of shorthand notations, called "Special Notations," for commonly encountered calculations that are otherwise a bit arduous to set up, and
- (3) you can use three "Special Operations" that can come in quite handy:
 - (a) you can round calculation values,
 - (b) you can employ a capability to remember and recall for use particular values in a calculation, and
 - (c) you can "pass" values from one row of a calculation to the next.

PLACING CONSTANTS INTO CALCULATIONS: THE CONSTANT CALCULATION LINE

Constant values can be placed into a calculation by employing a special type of Calculation Line. Like other Calculation Lines, it begins with a Control Code followed by a Sequence Number. These two elements are then followed by a special symbol that designates both that constants are being placed into the calculation and what, exactly, should be done with them. One such Calculation Line might look like this:

ex	999999	999999	999999
ac1	+	+	=
ac2/			4.5

In the above example, the Calculation Line beginning with "ac2/" is the one that introduces a constant into the calculation. It tells T/Maker to divide by 4.5 the value residing in column three (the initial value of column three was generated, of course, by the calculation prescribed by "ac1").

It is the "/" following the "ac2" that signals T/Maker a constant is going to be introduced and what is to be done with the constant. As noted already, in this case, the "/" tells T/Maker to divide the value of the third column by 4.5.

Only numerical values may appear in this sort of Calculation Line. Constants cannot be mixed in with operators or terminator symbols. In effect, though, a constant figure acts as its own terminator symbol--the result it generates is placed into the appointed column as if it had been dumped out by a terminator.

Notice in the example above that the constant is placed in the zone defined by the model number of the example line, but the level of accuracy in the associated model number and the entered constant are not the same. This is all right. T/Maker will not reformat constants according to the model number. Thus it is possible to use a precision for a constant not consistent with the column's model number.

More than one constant can be entered on such a Calculation Line. In the case below, for example, the Calculation Line would cause the value in the first column to be multiplied by 30 and the value in the third column to be multiplied by 365. The values of the second and forth columns would not be altered, however, because a constant operation is never performed when the constant given is either a blank or a zero.

ex	999999	999999	999999	99999
ac1*	30		365	0

The table below shows the various special symbols that can be used after the Sequence Number and describes the results each produces.

SPECIAL SYMBOLS FOR CONSTANTS

<u>Symbol</u>	<u>Result</u>
=	The column is set equal to the constant.
+	The constant is added to the value for the column.
-	The constant is subtracted from the value for the column.
*	The value for the column is multiplied by the constant.
/	The value for the column is divided by the constant.
%	The value for the column is divided by the constant and the result is multiplied by 100.
#	The value for the column is multiplied by the constant and the result is divided by 100.
max	The maximum of the value for the column and the constant becomes the new value for the column.
min	The minimum of the value for the column and the constant becomes the new value for the column.
^	The value for the column is raised to the power of the constant.

Tip: In situations where many different constants are used in different ways, it is often advisable to "store" all the constants so that they may be "fetched" when needed rather than having to use constant Calculation Lines. One of the example tables at the end of the discussion of Compute shows such an approach. (Store and Fetch are described later.)

An Example of the Use of Constants:

Suppose that Charlie wanted to calculate the pay for his employees on a particular job. Suppose that employee Ron makes \$9.35 per hour and John \$12.00 per hour. Ron worked 10 hours; John, 6 hours. Their pay could be calculated like this:

	Rate	Hours	Pay

ex	99.99	99.9	999.99
ac1	+	*	=
+	Ron	9.35	10.0
+	John	12.00	6.0
+	Sonny		

Now, suppose the weather was hot and Charlie hired Sonny to provide cold lemonade for Ron and John. He paid Sonny \$2.00 for every man-hour that Ron and John put in.

In fact, there are a number of ways to set up this calculation, but we will use one that employs a constant calculation:

	Rate	Hours	Pay

ex	99.99	99.9	999.99
ac2	+	*	=
+	Ron	9.35	10.0
+	John	12.00	6.0
jc1=		2.00	
=	Sonny	2.00	16.0
			32.00

Let's interpret the approach this table took. First of all, notice that we want to calculate Sonny's pay by multiplying his \$2/hour payrate by the total of Ron's and John's hours. It's an easy matter, of course, to calculate the sum of R and J's hours, simply by means of a Vertical Calculation Strip for the job.

But just such a Strip will also place a the total of R's and J's pay rates in the place where Sonny's \$2 pay rate should be. Therefore it is necessary that we override the Strip's calculation for Sonny's pay rate with a special constant value.

This was done by introducing the Calculation Line that begins with the code "jcl=" and sets the value in column one equal to 2.00.

Now, let's walk through this calculation exactly the way T/Maker does it. Remember that T/Maker proceeds one row at a time, from the top of a table to the bottom.

First, T/Maker goes to the first row of values and asks, "What calculations apply to this row?" Since T/Maker has so far "seen" only the Calculation Line multiplying rate times hours (Line "ac2"), it will apply that calculation to Ron's rate and hours, coming up with the result of 93.50, which is placed in the third column. Next, the same calculation is made for John's row of values.

T/Maker now encounters the Calculation Line beginning with "jcl=" and takes it into consciousness.

Next, T/Maker moves on to Sonny's row of values. It asks, "What calculations am I supposed to do?"

It sees three. The Vertical Calculation Strip tells it to compute the sums of the columns on this row. The "jcl" tells it to place a constant value of 2.00 in the first column of the row. And, finally, the "ac2" Calculation Line tells it to multiply the value presently occupying the first column by the value found in the second column, dumping the result into the third column. That is precisely the order in which T/Maker goes about the calculation of Sonny's row of values --first, Strip calculation; second, the calculation called for by the lowest Sequence Number; and, third, the calculation called for by the next higher Sequence Number.

Now, we have the result we wanted. Sonny should be paid \$32.00.

Converting Whole Tables to Percentage Figures:

This method for placing constants into calculations provides a quick and easy way to convert absolute figures into percentage figures. For example, consider the table below.

	X	Y	Total
ex	9999	9999	9999
uc1	+	+	=
+	A	4	2
+	B	5	6
+	C	6	9
=	Total	15	17
			32

Suppose that we wanted to percentage the figures in the three columns to their respective bases in the "Total" row.

Well, all we need do, then, is place a "uc1%" Constant Calculation Line under the example line, with the column totals inscribed in their respective columns, like this:

	X	Y	Total
ex	9999	9999	9999
uc1%	15	17	32
+	A	4	2
+	B	5	6
+	C	6	9
=	Total	15	17
			32

What, then, does this Calculation Line tell T/Maker to do? Well, this Line tells T/Maker that when it encounters a data value in the first column, it should divide that value by the specified constant (15 in this case), and then multiply the result by 100.

This Line also tells T/Maker that when it encounters a data value in the second column, it should divide that value by the specified constant (17 in this case), and then multiply the result by 100.

Finally, this Line tells T/Maker the same thing for values in the third column: data values in that column should be divided by the specified constant (32), and then the result multiplied by 100.

Invoking Compute would now produce the following result:

	X	Y	Total
ex	9999	9999	9999
uc1%	15	17	32
+	A	27	12
+	B	33	35
+	C	40	53
=	Total	100	100

Notice, by the way, that a "uc" ("usually compute") and not an "ac" ("always compute") Calculation Control Code was used in this operation. Why?

Once more, the reason lies in the time-order by which T/Maker proceeds through a calculating assignment. Remember that when T/Maker reaches a row that has a terminator symbol in its associated Vertical Calculation Strip, it computes these Strip values first, before proceeding to do whatever horizontal calculations as might apply.

Well, then, if we had used an "ac" code in the calculation above, T/Maker would have summed up the already percentaged values in the three columns and placed this sum (i.e., "100") at the base of each of the three columns in the total row. Next, following the instruction of the "ac" Calculation Line, it would have divided each of these 100's by the constant values associated with each column. Thus, we would not have gotten the result we wanted in the Totals row. By using the "uc" code, on the other hand, the percentaging calculation is not carried out on the final row. Instead, the totals presented there are simply the sum calculated by the Strip.

SPECIAL NOTATIONS

Special Notations form another category of Restricted Calculations. Special Notations provide the user with shorthand methods for doing a variety of commonly encountered calculations. Ordinarily, these calculations could have been done using the symbols of General Calculations, but Special Notations will save time and reduce the chances of mistakes, too.

Like other Calculation Lines, Special Notations begin with a Control Code and Sequence Number. Unlike other Calculation Lines, though, Special Notations do not have terminator symbols at the ends of lines of mathematical operators. One consequence of this difference is that a Special Notation line must always be put on a Calculation Line by itself.

SPECIAL NOTATIONS

cum+ + + +	Cumulates Values
chg+ + + +	Computes Absolute Changes
pch+ + + +	Computes Percentage Changes
mvr+ + + +	Moves Values to the Right
mvl+ + + +	Moves Values to the Left
pct + + +	Column Percents
- * + grw	Computes Growth Rates (1st Method)
- + / grw	Computes Growth Rates (2nd Method)
prj + + +	Computes Compounded Projections
inc+ + + +	Computes Straight Line Projections
dis net + +	Computes Net Present Values

The user should note that in the following pages each one of these Special Notations is described and an example of its use presented. Most of the exemplary calculations that follow have been presented in a common format:

```
ex      9999    9999    9999    9999    9999  
+      [data row . . . . . . . . . . ]  
jcl      [the Calculation Line containing the Special Notation]  
=      [row of derived values]
```

The Vertical Calculation Strip in this calculation causes the bottom row (of derived values) to be initially set equal to the data row. The Calculation line will then be applied to the bottom row. Thus, this form can show us the specific impact of each Special Notation.

CUMULATE VALUES

		Jan	Feb	Mar	Apr	Total
ex		9999	9999	9999	9999	9999
+	Values	1	2	3	4	10
jcl		cum+	+	+	+	
=	Cumulative	1	3	6	10	10

The special symbol "cum" cumulates the values in a row. The symbol "cum" can be put in any column of the line. Plus signs, however, must be put in the columns that will be cumulated. Columns with no associated plus signs will not be affected by the Special Notation. The cumulation proceeds from left to right. The leftmost column containing a plus sign remains unchanged.

ABSOLUTE CHANGES

	Jan	Feb	Mar	Apr
ex	9999	9999	9999	9999
+	Values	1	5	10
jcl	chg+	+	+	+
=	Changes	0	4	5
				10

The special symbol "chg" initiates a "change" calculation. The symbol "chg" can be put in any column of the Line. Plus signs, however, must be put in the columns that will be calculated. Columns with no associated plus signs will not be affected by this Special Notation. The "change" result placed in a given column is the difference between that column's value and the value found in the column to its left. The leftmost column is set to zero.

PERCENTAGE CHANGES

	Jan	Feb	Mar	Apr
ex	9999	9999	9999	9999
+	Values	1	2	3
jcl	pch+	+	+	+
=	Percent Change	0	100	50
				33

The special symbol "pch" initiates a percentage change calculation. The symbol "pch" can be put in any column of the Line. Plus signs, however, must be put in the columns that will be calculated. Columns with no associated plus signs will not be affected by the Special Notation. The "percent change" result is calculated in three steps: the first step calculates the difference between the value in a given column and the value found in the column to its left. The second step takes this intermediate result and divides it by the value found in the column to the left. The third multiplies this result by 100. The leftmost column in a row is always set to zero.

MOVING A COLUMN TO THE RIGHT

		Jan	Feb	Mar	Apr
ex		9999	9999	9999	9999
+	Values	1	2	3	4
jcl		mvr+	+	+	+
=	Moved Values	0	1	2	3

The special symbol "mvr" initiates a shift of column values to the right. The symbol "mvr" can be put in any column of the Line. Plus signs, however, must be put in the columns that will be shifted. Columns with no associated plus signs will not be affected by the Special Notation. The leftmost column is set to zero.

MOVING A COLUMN TO THE LEFT

		Jan	Feb	Mar	Apr
ex		9999	9999	9999	9999
+	Values	1	2	3	4
jcl		mv1+	+	+	+
=	Moved Values	2	3	4	0

The special symbol "mv1" initiates a shift of column values to the left. The symbol "mv1" can be put in any column of the Line. Plus signs, however, must be put in the columns that will be shifted. Columns with no associated plus signs will not be affected by the Special Notation. The rightmost column is set to zero.

COLUMN PERCENTS

	Jan	Feb	Mar	Total
ex	9999	9999	9999	9999
jcl	+	+	+	=
+	Values	1	2	3
jcl	+	+	+	+pct
=	Percents	17	33	50
				100

The special symbol "pct" percentages a row of figures to a designated base. The symbol "pct" is placed in the column that represents the base. Plus signs are put in the columns to be percentaged to that base. Columns with no associated plus signs will not be affected by the Special Notation. Note that the column with the "pct" symbol may also contain a plus sign--the result of this particular percentaging must, of course, be 100.

GROWTH RATES (1ST METHOD)

	Jan	Feb	Mar	Apr	Growth Rate
ex	999.9	999.9	999.9	999.9	999.9
jcl	-	*	*	+	grw
+	Values	100.0	110.0	121.0	133.1
					10.0

The special symbol "grw" initiates the calculation of a growth rate. A minus sign is put in the column representing the initial period. A plus sign is put in the column representing the final period. Asterisks are put in the columns representing the intervening periods. "Grw" is put in the column where the growth rate is to be placed. No special order is required for the placement of any of the symbols involved. For example, the minus sign could appear to the right of the plus sign, if your table happened to be set up that way.

The growth rate calculated in this manner is the percentage figure which if applied to the value for the first period in compounded fashion would yield the value for the final period after the specified number of periods. Because intermediate values do not enter directly into the calculation the asterisks can actually be placed anywhere on the Calculation Line. What is important, though, is that the right number of asterisks are used.

	1970	1975	Annual Growth Rate	
ex	999.9	999.9	999.9	
jcl	-	+	grw	****
+ Values	100.0	120.0	3.7	

GROWTH RATES (2ND METHOD)

	1970	mid- 1975	Periods	Annual Growth Rate
ex	999.9	999.9	999.9	999.9
jcl	-	+	/	grw
+ Values	100.0	120.0	5.5	3.4

The second method for computing growth rates allows you to specify an elapsed period that may include a fractional part. The elapsed time between the initial period and the final period is specified directly, by means of a value placed in the column falling under the slash sign. Of course, growth rates for whole-number intervals may be calculated by this method, too.

COMPOUNDED PROJECTIONS

	Growth Rate	1990	1991	1992	1995
ex	999.99	9999	9999	9999	9999
jcl	prj	+	+	+	+++
+ Values	10.00	100	110	121	161

The special symbol "prj" initiates the calculation of a compounded projection. The "prj" symbol is placed in the column that contains the percentage rate to be used for the projection. Plus signs are placed in the columns that are to be projected. The projection uses as its initial value the leftmost column that is both (1) associated with a plus sign and (2) contains a non-zero value.

Other columns are then calculated from left to right. Calculated figures are bolded in the above table. Notice the trick used to jump to the calculation of the projection for 1995.

STRAIGHT LINE PROJECTIONS

	Growth Increment	1990	1991	1992	1995
ex	999.99	9999	9999	9999	9999
jcl	inc	+	+	+	+++
+ Values	10.00	100	110	120	150

The special symbol "inc" calculates a straight line projection. This symbol is placed in the column containing the value of the absolute increment to be added for each period. Plus signs are placed in the columns that are to be projected. The projection uses as its initial value, the leftmost column which is associated with a plus sign and contains a non-zero value.

Other columns are then calculated from left to right. Calculated figures are bolded in the above table.

NET PRESENT VALUE

	Discount Rate	Net Value	1980	1981	1982
ex	999.9	99,999	99,999	99,999	99,999
jcl	dis	net	+	+	+
+	Values	10.0	15	-185	110
					121

The special symbol "net" initiates the calculation of a net present value. The result of the calculation is placed in the column associated with "net".

The value associated with the "dis" column is a percentage discount rate. The net present value is calculated as the sum of the discounted values corresponding to each of the plus signs. The discounted value for the n-th plus sign (counting from left to right) is computed by the following formula.

$$\text{undiscounted value} / [1 + (\text{discount rate} / 100)]^{(n - 1)}$$

In situations when it is desirable to start discounting from the first value (i.e., using "n" in the above formula instead of "[n - 1]"), the first plus sign can be placed under a dummy column that has a zero value. Putting more than one plus sign under a column can be used to achieve the effect shown below.

	Discount Rate	Net Present Value	First Year	Next 5 Years
ex	999.9	99,999	999,999	999,999
jcl	dis	net	+	+++++
+	Values	15.0	341	-1,000
				400

SPECIAL OPERATIONS

The Restricted Calculations described in this section are comprised of three kinds of operations--one for rounding calculation values, the second for storing and fetching values in a calculation, and the third for passing values from one row of a table to the next. Each operation is invoked by means of a single symbol, which can be placed on a Calculation Line by itself or mixed in with the symbols of a General Calculation.

"Special Operations" symbols cannot be placed on a Calculation Line containing a Constant Calculation Line or the Special Notations described in the preceding section. More than one Special Operation symbol on a Calculation Line is okay, though.

ROUNDING

As we have learned, every table has an example line. And, as it happens, not infrequently that example line will round off the results the calculation has generated. For example, consider this simple example:

ex	999	999	999
acl	+	/	=
+	100	3	33

In this calculation the fact that we have specified a whole-number model number for the third column--where the result of the division is dumped out--means that the T/Maker automatically rounds the result to 33. (T/Maker's automatic rounding procedure, by the way, is .5 rounds up.)

But T/Maker has not blythly thrown away the accuracy of this calculation but instead has merely presented the rounded result that was requested by the model number. A much more accurate result stays safe inside the T/Maker "calculator."

T/Maker's retention of this greater accuracy for calculation purposes is easy to demonstrate. For instance, in the example below we have added two Calculation Lines to the division problem described just above. The first additional Calculation Line simply picks up the result presented in Column C and dumps it out--under a more precise model number--in Column D. The second additional Calculation Line also picks up the result presented in Column C and dumps it out--this time under a much more precise model number--in Column E.

COLUMNS

	A	B	C	D	E
ex	999	999	999	99.9	99.999999
ac1	+	/	=		
ac2			+	=	
ac3			+		=
+	100	3	33	33.3	33.333333

As you can see, even though the results reported in Columns D and E were merely picked up from Column C, where only whole-number accuracy was reported, the final two columns fully report the accuracy of their associated example lines. T/Maker, in other words, is happy to present rounded numbers but keeps highly accurate numbers in its calculator nonetheless.

As it happens, this characteristic is generally desirable because it means that T/Maker uses the most accurate numbers in the calculation of equations.

On the other hand, this T/Maker characteristic can occasionally cause trouble. For example, consider the table below:

COLUMNS

	A	B	C	D	E
ex	999	999	999	999	999
ac1	+	/	=		
ac2			+	*	=
+	100	3	33	2	67

In this table the first Calculation Line divides 100 by 3, dumping the result into Column C. The second Calculation Line picks up the result of this division, multiplies it by 2 and puts the result into Column E.

But notice that T/Maker seems to have made a mistake: 2 times 33 is reported as 67. Of course, the reason for this is that T/Maker was working with a much more accurate version of the result placed in Column C (to be exact, 33.3333333333). Thus, the result the T/Maker calculator actually generated was 66.6666666666. This, in turn, was correctly rounded to 67 for presentation in Column E.

There may be times you will want the results of a calculation to reflect the values actually presented in a table rather than the underlying, more accurate values in T/Maker's calculator. In other words, you may want to round the T/Maker's "calculation" values in conformity with its "presentation" values. This is easy to do.

All you need do is add the instruction "rnd" to the appropriate column. This may be done on a Calculation Line of its own or mixed in with another Calculation Line above the Calculation Line that uses the value. Below, we have added a new Calculation Line:

COLUMNS

	A	B	C	D	E
ex	999	999	999	999	999
ac1	+ /		=		
ac2			rnd		
ac3			+ *		=
+	100	3	33	2	66

Putting "rnd" on the second Calculation Line below after the terminator symbol shows that "rnd" will round values to the accuracy of the model number immediately after the value for column D is calculated.

COLUMNS

	A	B	C	D	E
ex	999	999	999	99.9	99.999999
ac1	+ /		=		
ac2			+ =rnd		
ac3			+ =		
+	100	3	33	33.3	33.300000

A Special Application for Rounding:

The example below illustrates a simple technique for rounding a value to the nearest 10 dollars.

	Amount	Nearest 10
ex	99,999	99,999
uc1	+ =	
uc2/		10
uc3		rnd
uc4*		10
+	Item 1	150
+	Item 2	130
=	Total	280

STORING AND FETCHING VALUES

T/Maker maintains a sort of pigeonhole-cabinet memory for individual numbers that can be used in any table. There are twenty-six slots in this "cabinet," into which numbers can be placed and from which numbers can be retrieved again whenever they are wanted. Each slot is identified by a single-letter name, from "a" to "z".

Putting a number into a slot is called "storing" it, and taking it out again is called "fetching" it. The codes for these operations are "st" and "ft". The name of the slot in question is designated by the third letter of the symbol. For example, "stm" means "store this number in slot m." "Ftx", on the other hand, means "fetch the number in slot x."

These codes can be placed on Calculation Lines by themselves or they can be mixed in with the symbols of a General Calculation. Like all Special Operations, they cannot be used in Calculation Lines that include Constants or Special Notations.

When the "store" operation is made, the value stored is the current value for the column associated with the store symbol. When the "fetch" operation is used, the value fetched is placed into the column associated with the fetch symbol.

Consider the table below, which uses storing and fetching for the simple purpose of presenting a total column to the left of the values used to calculate it.

	Total	1980	1981	Total
ex	999,999	999,999	999,999	999,999
ac1		+	+	= sta
ac2	fta			
+	Item A	7	5	2
+	Item B	15	4	11
				15

The first Calculation Line computes total values made up of 1980 and 1981 figures and places the results in the fourth column. This result is then stored into slot "a". The second Calculation Line fetches the value in slot "a" and places it into the first column. ("Fta" could not have been put on the first Calculation Line, since to fetch something it must first have been stored.)

In the table below, store and fetch are used to produce a figure representing the fraction that total unearned income represented when compared to total earned income. Note that a value stored from one row can be fetched into service in another row.

		1980	1981	Total
ex		99,999.99	99,999.99	99,999.99
ac1		+	+	=
+	Salary	18,000.00	20,000.00	38,000.00
+	Consulting	5,000.00	8,000.00	13,000.00
jc2				ste
=	Total Earned Income	23,000.00	28,000.00	51,000.00
+	Dividends	2,000.00	500.00	2,500.00
+	Interest	2,000.00	4,000.00	6,000.00
jc2				stu
=	Total Unearned Income	4,000.00	4,500.00	8,500.00
jc1		fte	ftu	
jc2		/	+	%
	Unearned As Percent			
+	Of Earned Income	51,000.00	8,500.00	16.67

Let's see what went on in the above example: Total Earned Income was stored into slot "e" by a single Calculation Line. Total Unearned Income was stored in slot "u" by a similar Calculation Line.

At the bottom of the table these two stored values were fetched back again and placed into the first and second columns. (Incidentally, the Calculation Line that generated this result used Sequence Number "1" again, which in effect cancelled the earlier Calculation Line employing that same Sequence Number.) Once the values were fetched, a division calculation was carried out in order to generate the desired result.

You might note, by the way, that the last row must have some symbol in the first seven positions or T/Maker will not recognize it as a row upon which to carry out calculations. Any symbol would have done the trick in this case.

It is interesting that the calculation carried out by the final two Calculation Lines might have been carried out a little more elegantly. Remember that store and fetch symbols can be intermixed with the symbols of General Calculations. Consequently, the division operator could have been placed after the "fte" symbol, underneath the model number in column one. This would have set the calculation value in column one to 51,000 when it was processed. Similarly, a plus sign placed after "ftu" would have had the effect of lodging 8,500 into the calculation value of the second column. Even though Compute's calculator carries out addition operations before it carries out division operations, T/Maker, nonetheless, will accept into its consciousness the information coming from a General Calculation on a left-to-right basis. Hence, the Calculation Line shown below would have been all that was necessary to carry out the desired calculation. Mixing store and fetch symbols within General Calculations can be a very powerful device.

ex	99,999.99	99,999.99	99,999.99
jcl	fte /	ftu +	%
	Unearned As Percent		
+ Of Earned Income	16.67		

When examining a calculation like the one above, one needs to remember that store and fetch operate a sort of shuttle service between the pigeonhole-cabinet and the values in the columns. This shuttle service runs independently of the "shovels" that scoop up column values for calculations. Fetching a value does not, itself, enter it into a calculation (i.e., shovel) but simply puts it into the column.

However, the shuttle service and the scooping shovels are synchronized on a left to right basis. Thus in the same column, one can shuttle in a value, scoop it up, shuttle in another value, scoop it up, etc. The example below would, again, have produced the same result for the third column!

ex	99,999.99	99,999.99	99,999.99
jcl	fte/ftu+%		
	Unearned As Percent		
+ Of Earned Income	16.67		

A Handy Feature of Fetching:

When a fetch is attempted from a slot containing no stored value, a zero is fetched. This fact can have useful consequences. For example, in the table below the use of "ftc", when applied to the first row, will fetch a zero. However, the use of "ftc" when applied to the second row will fetch the value stored from the first row.

	Amount	Cumulative
ex	999,999.99	999,999.99
uc1	+	ftc + = etc
+ Row 1	1.00	1.00
+ Row 2	2.00	3.00
+ Row 3	3.00	6.00

Fetching zeros (in conjunction with the use of an "all blanks" Zero Values Line) can provide a convenient way to suppress the printing of meaningless values. Blanked-out meaningless values can much enhance the clarity of tables. And, this technique might be easier to do than modifying example lines in order to produce invisible columns (see Advanced Example Line). The table below demonstrates the approach:

	Item	Quantity	Price	Total Price
ex		999999	999.99	99,999.99
zv				
uc1		+	*	=
rc2		ftz	ftz	
+ Item A		10	10.00	100.00
+ Item B		15	15.00	225.00
= Total				325.00

Another handy utility for store and fetch is that of storing all of the original data values required in a calculation for use later on. This tactic can free up the table's columns for intermediate uses.

Perhaps this utility is best conveyed by means of an example. Suppose that you wanted to set up an equation for finding the results of a reasonably arduous formula. Suppose, too, that you wished to produce a table that kept the original values of the computation intact.

For the sake of illustration, let us say that you wanted to find the square root of the quantity

$$A^3 + 2B^2 + 4C$$

for the following sets of values for A, B, and C:

{A, B, C}

(3, 5, 17)
(5, 8, 19)
(4, 7, 15)
(2, 4, 18)

Now, how are we going to do is this: Well, let's solve it, first, for the first set of values, {3, 5, 17}.

	Square Root Of			
	A	B	C	$A^3 + 2B^2 + 4C$
ex	9999	9999	9999	99999.999
ac1	sta	stb	stc	
ac2	+***	+**		
ac3*		2	4	
ac4	+	+	+	sqr
ac5	fta	ftb	ftc	
+	3	5	17	12.042

Let's interpret this table: The first Calculation Line immediately stores the values of the first value row into three pigeonholes, a, b, and c.

The next Calculation Line cubes the value in the first column and dumps out this cube in the first column as a result. Next, this same Calculation Line squares the value in the second column and dumps out this square in the second column as a result.

The third Calculation Line is a one for introducing constants. This Line leaves the value in the first column unaffected, but it multiplies the value in the second column by the constant, 2, and it multiplies the value in the third column

by 17.

The next Calculation Line (ac4) picks up all three column values in a General Calculation. This calculation sums the values and takes the square root of the sum, which result is dumped into column 4.

Now, our computation has come to an end.

As a final step, the bottom Calculation Line (ac5) fetches the three values we started with and puts them back into columns one, two, and three. Thus, the solution to the equation and the values for which the equation was solved will appear next to each other in the final table.

All we need do in order to solve for all of the sets of values we first mentioned is add more rows of values, like this:

	Square Root Of			
	A	B	C	$\sqrt{A + 2B + 4C}$
ex	9999	9999	9999	99999.999
ac1	sta	stb	stc	
ac2	***=	**=		
ac3*		2	4	
ac4	+	+	+	sqr
ac5	fta	ftb	ftc	
+	3	5	17	12.042
+	5	8	19	18.138
+	4	7	15	14.900
+	2	4	18	10.583

PASSING VALUES

In some tables it is convenient to pass along a value from one row to the next row. As we've seen, Store and Fetch symbols could be used to accomplish this. But, there is an easier way, using the "pas" code, which stands for "pass". "Pas" simply passes the value present in the column to the same column-site in the next row.

Values can be passed even to rows that have been derived by a Calculation Strip. (Keep in mind, though, that in this case the "passed" value will override whatever value might have appeared in the column as a result of the Strip's calculation. Remember

that T/Maker calculates Strips first and then does other calculations, including passes.)

Below, "pas" is used to solve the checkbook problem:

	Deposit	Check	Balance
ex	999,999.99	999,999.99	999,999.99
zv		+	-
uc1			+=
ac2			pas
+ Opening Balance			1,200.00
+ Salary	1,300.00		2,500.00
+ Car Payment		200.00	2,300.00
+ House Payment		580.00	1,720.00
= Total	1,300.00	780.00	1,720.00

Here we can see that the first Calculation Line provides for the adding-in of deposits (first column), the subtracting-out of checks (second column), and computation of the resulting balance (third column). Let's take a closer look at what is going on in the third column.

The plus sign in the third column of the first Line tells T/Maker to add into its calculator whatever value appears in this column. This provides for the inclusion of the past balance into the current balance figure that will be calculated. Then, the equal sign that follows the plus sign tells T/Maker to present a result, the current balance, in the third column.

Now, the second Calculation Line takes the value it finds in the third column (the result just calculated) and passes it straight down to the row below. So you see, when the next row of values is considered by the first Calculation Line, that row starts out with the old balance in the third column--just where you want it! When the calculation is complete the new balance will replace the old.

Notice, incidentally, that no value is passed to the first row.

Another example of "pas" is presented below. This one does a vertically-arranged compounded projection.

		Item A	Item B	Item C
ex		999,999	999,999	999,999
uc2		pas	pas	pas
+	1982	200	100	50
uc1*		1.10	1.15	1.20
+	1983	220	115	60
+	1984	242	132	72
+	1985	266	152	86
+	1986	293	175	104

Let's consider what is going on in this table. First of all, notice that there are five lines of values (one each for the years 1982-1986) but the top values line, for 1982 values, is separated from the others. That's because 1982 values will provide the base values for the projections.

T/Maker approaches this calculation as follows. First, it encounters the Calculation Line "uc2", just beneath the Example Line. T/Maker sees that it should pass the values in the first line of data down to the next values line.

So, think of the 1983 values line as starting out with the 1982 values in it.

Next, T/Maker moves along and it finds the Calculation Line "uc1". This Line tells T/Maker to multiply the values in columns by constants it has specified (1.10 for the first column, 1.15 for the second column, and 1.20 for the third column). So, when T/Maker hits the 1983 row of values it will multiply the values it initially finds in that row (remember, these will be 1982 values) by these constants. The products of those multiplications will now replace the initial values in this row.

Next, T/Maker will look for the next calculation it is supposed to carry out on the 1983 row. It finds the Calculation Line "uc2", telling T/Maker to pass the 1983 values to the next row.

This, of course, starts the whole cycle over again, completing the projection we wanted to calculate.

Notice that the first row of values is unaffected by the "pas". This is always the case, because "pas", after all, is inherently an activity that affects "the next row down".

ITERATION

When the Compute command is given to T/Maker, its calculator begins with a clean slate. In other words, no values are stored or ready to be passed, no calculation lines or strips are active, and so on. It will take the particulars of a new calculation problem to begin to fill up these various areas with values.

There may be times, though, when you would like T/Maker to begin a calculation with certain values already in mind.

As it happens, this can be accomplished by using the Compute command in succession on the same table. Like this:

WHAT NEXT? COMPUTE C C C

Notice that subsequent commands can be abbreviated to the letter C.

Giving the Compute command two or more times in succession serves the purpose of carrying over values from the previous calculation into the next pass at the table. Like a loom with a memory, subsequent sweeps of the Compute command bring along with them the remembrance of results that may be of use.

This very simple example illustrates the point:

```
ex      9999    9999
=
+
      1      3
+
      2      4
```

Here, the command "Compute C" would send T/Maker's calculator through the table twice. On the first pass, the calculator would add 1 and 2 in the first column and add 3 and 4 in the second column. But because T/Maker works from top to bottom the first pass of the Compute command will have no place to deposit the result. On the second pass of the calculator, then, the terminator symbol will cause T/Maker to dump out the results of the addition in the top row.

The effect is the same as stacking a table under itself, like this:

```
ex      9999  9999  
=  
+          1    3  
+          2    4  
  
ex      9999  9999  }  
=          } second occurrence of the table  
+          1    3  }  
+          2    4  }
```

The result of this processing would be simply the table below:

```
ex      9999  9999  
=          3    7  
+          1    3  
+          2    4
```

The iterative property of Compute can be quite useful for a variety of problems. In the table below, for example, Compute's first pass calculates a column total. The second pass in turn is used to convert absolute figures to percents.

	Dollars	Percents
ex	99,999	99,999
+	Item A	100
+	Item B	250
+	Item C	150
jcl		stt
=	Total	
uc1		+ ftt /%

Whenever this feature of T/Maker's calculator is desired, all iterations must be specified in a single command (for example, "compute c c c c").

OTHER TOPICS

Print Design Commands:

The Compute command ignores print design commands. Thus design commands pertinent to the final presentation of the table when printed may be used as needed. For example, a table might be started with the lines shown below.

```
stop
```

```
-----  
JOB ESTIMATES  
-----
```

Item	Quantity	Price	Total
------	----------	-------	-------

```
-----
```

```
end
```

The use of a Top Routine here insures that a printed table that extends beyond one page will have the same headings on each page. (See the last example of the Word Processing Section and the Clean command for examples and advice on placing design commands in a table.)

Advanced Example Line:

INVISIBLE COLUMNS

If a model number on an Example Line begins with a comma (i.e. ","), the number for that column will be blanked out. It is sometimes useful to have such invisible columns in a table for the purpose of holding intermediate results. Many calculations are easier to specify through the use of invisible columns. Also, this technique is sometimes used to avoid the presentation of meaningless columns for one or more rows.

Since such columns are always presented as blanks, it makes no difference whether or not the model number really provides a valid "model" of the numbers in the column.

Three
Invisible Columns

<----->

ex 999,999 999 ,,, , ,999 999,999

The reason for picking a narrow or wide model for an invisible column has to do with the width of the symbols one intends to place under the model and not the size or precision of the numbers to be used in the column.

CHANGING THE EXAMPLE LINE

The example line in a table can be changed at any point by giving a new one. Thus, a row of averages could be printed with more precision than the rows that are used to calculate it. An example line, of course, always applies to all rows beneath it until a new example line is introduced.

When the example line is changed, there needs to be an assumption made as to what old column is what new column, since calculations below the new example line may make use of values above it. The association is made on the basis of order (e.g. the third column on the first example line is assumed correspond to the third column the next example line).

ex 9.9 99999 9,999 999,999
ex 999 999 99.99 99.99

It is good practice not to change the number of model numbers (or columns) on an example line even though you may change the model numbers.

Incidentally, if an example line with no model numbers on it is introduced, then any following Calculation Lines, Calculation Strips, or even text in positions one through seven is ignored entirely. Compute will continue calculating again only after an example line with at least one model number on it is encountered.

Another reason for changing the example line, is demonstrated in the table below.

	78	79	80	81	GRW	82	83	TOT	AVR
ac1	-	*	*	+	GRW				
uc2				+	prj	+	+		
ac3	+	+	+	+		+	+	=	
ac3	+	+	+	+		+	+		=

-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
Item	1978	1979	1980	1981	Growth	Rate	etc.	-----	-----
ex	999,999	999,999	999,999	999,999	999.99	999.99)	-----	-----

When a table is particularly wide, it can be helpful to specify Calculation Lines in a more compact format. This is accomplished in the above table by using a condensed example line for coding the calculations, and then using a spread out example line for the actual table. Notice how characters other than nines are used in the first example line to help remember which column is which.

In fact, any other characters (except commas and periods) may be used to form the model number in place of the nines.

Placing Numbers Below Example Lines:

When numbers are placed in a table, they must be placed in the zone defined by the appropriate model number. However, they need not be justified in any particular way nor must they conform to their model number on the example line. By far, the easiest way to put in a lot of numbers is to set tabs at the leftmost position of each model number (see Set All Tabs in the editor for a quick way to do this) and put the numbers in left-justified.

Positions not under a model number may be used for any type of text- row names, columns of text, footnote references or special symbols next to a number, etc.

-----	-----	-----	-----	-----
Item	Quantity	Price	Total	
-----	-----	-----	-----	-----
ex	999,999	999.99	99,999.99	
Zones:	<---->	<---->	<---->	
Diskettes	500	5.9	(a	

Compute will, itself, format numbers according to their models.

Decimal points must be included if the number has a fractional part, and negative numbers must be preceded by a minus sign. No spaces or characters other than a comma should be included. Below are some examples.

ex	999,999.99	
	12.3	RIGHT.
	158	RIGHT.
	-1,234	RIGHT.
	587	WRONG. The number starts one position too soon. The number will be interpreted as 87 and the "5" will be treated as text.
	56.6	WRONG. The number starts one position too late. The number will be interpreted as 56 and the "6" will be treated as text.
	12 3	WRONG. The number contains a space. It will be treated as 12.
	17-	WRONG. The minus sign must come first. This number will be treated as 17.

Tip: When editing or changing numbers and text in a table, be leary of using character insert or character delete operations as these shift the position of following elements on the line. After this mistake is made a few times, one learns to adjust for it.

Also, when changing a number in a table, it is useful to know that Compute scans numbers from left to right and stops scanning as soon as an illegal character is found. Thus, for example, to set a number to zero, one can just overwrite the first character in it with an illegal symbol, say "z". To change the number 875 to 4, one could type the "4" over the "8" and a space over the "7". Compute would present these numbers correctly at its next use. Use of these techniques makes it much quicker to revise the numbers in a table.

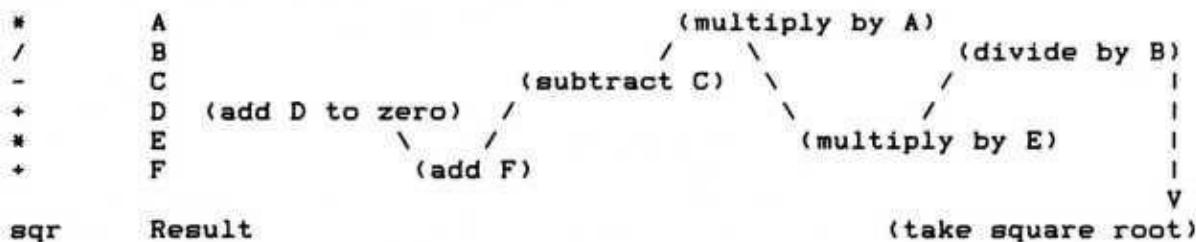
A Few Words On Vertical Calculation Strips:

ENTERING STRIPS

When entering Strips, an undeniable temptation is to enter all symbols proceeding from top to bottom. A better approach, however, is to enter one strip at a time and to enter the symbols for each strip in the order they will be processed. In other words, enter all the plus signs first, then all the minus signs, then all the multiplication signs, then all the division signs, and finally, the terminator symbol.

Using this approach requires some extra zipping of the cursor up and down, but it makes it possible to keep a running calculator in mind which simulates the actual process used by Compute. The same approach can be used to go back the next day and grasp the intention of a calculation (i.e., start examining the plus signs first, etc.).

The diagram below suggests the order in which the symbols might have been entered and the running calculator that might have been kept in mind.



INVISIBLE ROWS

Not all calculations can be performed in a single strip. Often, it is necessary (and helpful) to use additional rows for the sake of holding intermediate values. These rows can be eliminated when printing in a number of ways. Consider the table below.

ex	999, 999
+	A
+	B
= /	.. A+B
+	C
+	D
=	(C+D)/(A+B)

Here, "A+B" is an intermediate result which is necessary only for the calculation of the final row and presumably should not be printed. Starting in position 8, this row begins with two periods. Thus, after using the Clean command on the table (see Clean command), this row will become a print design comment line (see Print Design Commands) and will therefore not be printed.

The Drop command could also be used to eliminate intermediate rows. For the case above, "drop .." would work.

Debugging Control Codes and Sequence Numbers:

Depending on the complexity of a problem, everyone from beginner to expert can have a table go awry because the wrong Calculation Lines are being applied to a row.

Here is a procedure which shows exactly which lines apply to which rows. Although it takes a few minutes to do, it is well worth the effort especially if one has spent five minutes looking for a problem in a table and cannot find it.

First, use a copy of the problematic table and erase everything to the right of the row titles and everything before the first example line (see Clip command). Next delete all example lines and use an example line that has at least as many columns as the highest Sequence Number used on any Calculation Line. A Zero Values Line with dashes should also be included, and an "ac0" equation which puts zeros into all the columns. (Hopefully, the table does not use Sequence Number zero itself.) The result might look like the next table.

	1	2	3	4	5	6	7	8
ex	99	99	99	99	99	99	99	99
zv	-	-	-	-	-	-	-	-
ac0	=	=	=	=	=	=	=	=
ac1								
uc2								
rc3								
uc5								
+	Row A							
sc1								
*	Row B							
uc8								
= +	Row C							
cc5								
+ *	Row D							
ac2								
+ #	Row E							
*	Row F							
max	Row G							

Now go back through the table and change any usage of "ac", "uc", "rc" or "jc" to a constant calculation that places the Sequence Number of itself into the column that corresponds to its Sequence Number. The result at this stage will look like the table below.

	1	2	3	4	5	6	7	8
ex	99	99	99	99	99	99	99	99
zv	-	-	-	-	-	-	-	-
ac0	=	=	=	=	=	=	=	=
ac1=	1							
uc2=		2						
rc3=			3					
uc5=				5				
+	Row A							
sc1								
*	Row B							
uc8=							8	
= +	Row C							
cc5								
+ *	Row D			2				
ac2=								
+ #	Row E							
*	Row F							
max	Row G							

When this table is computed and cleaned, it produces the result below. Notice that each row now lists the Calculation Lines which apply to it.

	1	2	3	4	5	6	7	8

Row A	1	2	-	-	5	-	-	-
Row B	-	2	-	-	5	-	-	-
Row C	1	-	3	-	-	-	-	-
Row D	1	2	-	-	-	-	-	8
Row E	1	2	3	-	-	-	-	-
Row F	1	2	-	-	-	-	-	8
Row G	1	2	3	-	-	-	-	-

If the table had contained any uses of "cancel compute" or "suspend compute" which applied to all Calculation Lines and not a specific Sequence Number, then the "ac0" Calculation Line should have been re-entered immediately after the line with "cc" or "sc".

	1	2	3

ex	99	99	99
zv	-	-	-
ac0	=	=	=
ac1=	1		
uc2=		2	
rc3=			3
+	Row A		
*	Row B		
cc			
ac0	=	=	=
=	Row C		

ERRORS AND LIMITS

From time to time, Compute may give an error message. These messages are supposed to be self-explanatory. When an error occurs, the top line on the screen is set to the line causing the error. Thus, one can immediately enter the editor to correct the error, and then try Compute once more.

Some minor problems, such as attempting to take the square root of a negative number or trying to print a number with a model number that is too small, are not diagnosed. Whenever an inappropriate calculation is performed, the result is simply set to zero and processing continues normally. A count of such "MINOR ERRORS" is printed by Compute at the top of the screen if any occur.

If you see the general error message "OUT OF SPACE", it means that your table is too big. You can reduce its size by eliminating anything in the table that will increase the number of free bytes (you can keep track of your progress in such reducing efforts by the information function in the editor). Deleting text or squeezing up the spaces between columns is just as effective as deleting rows, columns, or calculations.

Compute permits a table to consist of up to 25 columns. Otherwise, most anything that the editor will permit in terms of size can be processed by Compute. The maximum number of Calculation Lines active at any time is 21 (numbered 0 through 20). The maximum number of symbols in Calculation Lines is approximately 200—it depends on the type of calculation specified. A constant in a Constant Calculation Line uses the equivalent of 4 symbols.

If the message "TOO MANY SYMBOLS OR CONSTANTS" appears try putting in a line with just "cc" (i.e., cancel all calculation lines) a bit above the line where the problem occurred. Below the "cc" line, re-enter the calculation lines that should still be active below.

HAND CALCULATOR PROBLEMS

ex		999,999.99
+ Salary	26,000.00	
+ Consulting	5,000.00	
+ Interest	600.00	
= + Taxable income	31,600.00	
+ Business expenses	2,500.00	
+ Depreciation	4,000.00	
= - Business deductions	6,500.00	
- Personal deductions	3,000.00	
+ = Net income	22,100.00	
- Bracket limit	20,200.00	
+ Bracket tax	3,273.00	
* Marginal rate	28.00	
# = + Total tax	3,805.00	
- Investment credit	100.00	
- Child care credit	400.00	
= Approximate tax due	3,305.00	

Here is an example of using Compute as a hand calculator. In such cases, the table usually has only a single column. Calculation strips are used to specify the operations required.

Notice that if you do not want to bother with row headings, then what you have to type is about the equivalent of what is required to solve the problem with a hand calculator.

Other columns could easily be added to try the same set of calculations on a different set of values while preserving the originals, because Calculation Strips always apply to all the columns in a table.

RECORD KEEPING

Utility Bills					
	Month	Phone	Electricity	Water	Total
ex		99,999.99	99,999.99	999.99	99,999.99
zv					
aci					
+	January	85.41	195.23		280.64
+	February	59.72	175.19	98.41	333.32
+	March	71.85	177.11		248.96
+	April	54.70	116.13		170.83
+	May	60.13	88.52	104.65	253.30
+	June	87.75	77.12		164.87
+	July	82.21	85.50		167.71
+	August	63.60	80.32	67.94	211.86
+	September	144.46	81.62		226.08
+	October	210.49	109.51		320.00
+	November	96.03	77.19	50.95	224.17
+	December	119.97	107.59		227.56
=	Total	1,136.32	1,371.03	321.95	2,829.30
avr	Average	94.69	114.25	26.83	235.78

Many records can be kept as simple tables.

A SIMPLE CASH FLOW MODEL

Cash On Hand					
	Jan	Feb	Mar	Apr	
ex	999,999	999,999	999,999	999,999	
zv					
+	House Mortgage	513	513	513	513
+	Other	800	825	850	900
= -	Total Expenses	1,313	1,338	1,363	1,413
+ =	Salary	1,900	1,900	2,100	2,100
+	Dividends			100	
= +	Total Income	1,900	1,900	2,200	2,100
+ =	Net Income	587	562	837	687
+ =	Current Bank Balance	1,540			
jcl		cum +	+	+	+
=	New Bank Balance	2,127	2,689	3,526	4,213

Here is a simple cash flow model. Additional rows can easily be inserted for other categories of expense or income. More columns may be added to widen the time frame.

If such a model is to be kept up to date, an easy way to make the change at the end of the month is to move the first column of numbers to become the last column of numbers. Then only the name of the month and the numbers for it need be changed in the last column. The "Current Bank Balance" would also require updating.

STOCK PORTFOLIO

CURRENT HOLDINGS					
	Stock	Shares	Price	Current Value	
ex		999,999	9999 9 9 ,	999,999.99	
zv				+ / =	
uc1				+ + =	
uc2		*		=	
+	Moon Tours Inc.	100	10 1/2	1,050.00	
+	Better Mousetrap Co.	75	121 1/8	9,084.38	
+	IBM	40	452 1/4	18,090.00	
+	Telegen	20	110	2,200.00	
ex		999,999		999,999.99	
=	Total	235		30,424.38	

Here is a basic stock portfolio which permits prices to be specified as fractions. Calculation Line 1 converts the fraction to a decimal value and places the result in an invisible column. Calculation line 2 adds the integer part of the price to the fractional part and multiplies this quantity by the number of shares.

The use of invisible columns (i.e. columns that can hold calculations, but do not print) often makes it much easier to specify the Calculation Lines required.

CALCULATING WITH DATES

GROWTH OF PROPERTY INVESTMENTS

Lot		Date Of Purchase	Purchase Price	Date Of Assessment	Assesed Value	Approx. Monthly Rate Of Increase
ex		99 99 99	999,999	99 99 99	999,999	999.9
uc1		sta stb stc		ste stf stg		
uc2*		12		12		
uc3/		30		30		
uc4		- - -		+ + +		=
uc5					+	/grw
uc6		fta ftb ftc		fte ftf ftg		
+	A	10/ 8/80	55,000	11/ 5/82	62,000	0.5
+	B	12/14/81	48,000	12/ 1/82	48,500	0.1
+	C	9/18/81	14,000	12/ 3/82	18,000	1.7
+	D	11/ 8/82	120,000	12/ 8/82	132,000	10.0

In this table, dates are used in calculations to derive an approximate monthly growth rate for some investments. Calculation Line 1 simply stores the dates. Calculation Lines 2 through 4 convert the dates to "absolute months" by multiplying the year by 12 and dividing the number of days by 30. The difference between the two dates in months is then placed in the last column.

Finally, the Growth Rate Special Notation is used to calculate the average growth of the investment on a monthly basis. The original dates are "fetched" back by Calculation Line 6.

While all months do not have thirty days, this technique will produce relatively accurate results so long as the time period is not too short.

A CHECKBOOK

```
=====
FIRST AMERICAN ACCOUNT
=====
Date To/For:      Type   Deposit   Check   Balance
=====
8-10          32-34    37-46    48-57
=====
ex           999,999.99 999,999.99 999,999.99
zv
uc1
ac2
+ Jan 01 Opening Balance           +     -     +=     pas
                                         3,687.00
+ Jan 03 C and P                 Utl      55.34  3,631.66
+ Jan 03 Safeway                  Fod      116.72 3,514.94
+ Jan 31 Salary                   Dep    2,345.11  5,860.05
= + January Total                2,345.11 172.06  5,860.05
+ Feb 10 Cash                    Gen      150.00  5,710.05
+ Feb 25 Cash                    Gen      250.00  5,460.05
= + February Total               400.00  5,460.05
= Grand Total                   2,345.11 572.06  5,460.05
=====
```

Here is an easy solution to keeping a checkbook that permits subtotals as needed. Notice that checks have been classified by "Type." This is done so that the Tally command can be used on the table. The command below would produce a summary of expenses by type.

```
TALLY IT 32 34 SUM 48 57 END
```

When the statement from the bank arrives, a quick way to check it is to delete the lines corresponding to checks and deposits which have not yet been processed by the bank. Then use Compute, the total should be the same as that shown by the bank. This stripped down file should, of course, not be saved.

When entering new checks, much time can be saved by using the Copy to the Buffer feature of the Editor. As you look through the list of new checks, put a copy of an old one that is to the same person (if you have it) into the buffer. Then, if these lines are inserted at the end of the list, only the dates and perhaps the amounts need to be changed.

DEPRECIATION TABLE

TAX DEPRECIATION CALCULATIONS						
Date Item	Purchase Price	Previous Depreciation	This Year's Depreciation	Amount	Percent	Amount
ex	99,999.99	99,999.99	999.999	99,999.99		
zv						
uc1		+	*		# rnd	
•.2						sta
•.3			+ fta + =			
rc9					ftz	
+	02/80 Computer	3,000.00	750.00	38.000	1,140.00	
+	05/80 Office	42,000.00	3,360.00	11.000	4,620.00	
+	07/80 Car	12,658.00	3,164.50	38.000	4,810.04	
+	03/81 Plotter	2,451.00		25.000	612.75	
=	Total	60,109.00	7,274.50			11,182.79

Here is a simple depreciation table. As the year progresses, new equipment can be added to the list, and Compute can be used to see the expected total amounts for the year end. Note that "rnd" is used to round the calculated depreciation. This is done to insure that the total of this column will be exactly the total of the numbers printed.

To start the table for the next year, "..." is replaced by "uc" and the Compute command is given to update the "Previous Depreciation." Equations 2 and 3 are then suppressed again by making them print comment lines (i.e., lines beginning with "..."). The allowable depreciation percentage must also be changed for the new year.

Individual calculations using the "jc" Control Code could be put ahead of particular rows to accomplish any special needs--sold items, etc.

Note the use of calculation "rc9" to force the total of the percent column to zero. "Fetching" a number for which nothing has ever been stored always yields a zero. This way to suppress the printing of a column often turns out to be easier than changing the example line so that the model number for the column begins with a comma.

PROJECTIONS BASED ON PAST DATA

	CAR COSTS					
	1980	1981	1982	Growth Rate	1983	1984
ex	9,999	9,999	9,999	999.9	9,999	9,999
ac1	-	*	+	grw		
uc2			*	prj	+	+
+	Fuel	870	980	1,105	12.7	1,245
+	Insurance	320	305	290	-4.8	276
+	Registration	20	20	20	0.0	20
+	Service	100	150	150	22.5	184
+	Repairs	10	200	250	400.0	1,250
=	Total	1,320	1,655	1,815	17.3	2,975
						8,161

In the above table, past data is used to make future projections. Calculation Line 1 computes the growth rate for all rows from the first three columns. Calculation Line 2 projects the 1982 value based on the observed growth rate. This calculation line is coded as "uc", which means it is not applied to rows derived from other rows. The only row in the above table which is calculated from others is the total row. Thus the figure "2,975" is not a projection of "1,185" at "17.3%," but rather the total of the figures above it.

Notice that making blind projections on past data has its pitfalls. Clearly, the growth rate for "Repairs" is unlikely to continue. Minimums and maximums could be imposed on the growth rates to insure that they fell within some reasonable range.

CUMULATIVE AND PERCENT CHANGE COLUMNS

STAFF CHANGES						
Year	Staff Hired	Staff Terminated	Net Increase	Total Staff	Percent Change	
ex	9,999	9,999	99,999	99,999	9999.9	
zv						
uc1						%
uc2						
+						
1971	20	2	18	18		
1972	18	3	15	33	83.3	
1973	25	5	20	53	60.6	
1974	31	9	22	75	41.5	
1975	21	11	10	85	13.3	
1976	30	14	16	101	18.8	
1977	25	18	7	108	6.9	
1978	21	17	4	112	3.7	
1979	14	12	2	114	1.8	
1980	15	13	2	116	1.8	
ex	9,999	9,999	99,999			
=	Total	220	104	116		

This table treats the problem of both a cumulative column and a percent change column. Stored value "s" is used to hold the total staff from the previous row.

When calculating the first row, "s" will contain a zero. Therefore, calculation "uc1" produces a division by zero (a mathematical error) which yields a result of zero (correct for the circumstances).

A TABLE WITH MORE THAN 25 COLUMNS

	1981												1982												Total	Growth	1983		
	J	F	M	A	M	J	J	A	S	O	N	D	T1	J	F	M	A	M	J	J	A	S	O	N	D	T1	(est)		
ex	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	Total	Growth	1983		
ac1	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	(est)				
ex														99	99	99	99	99	99	99	99	99	99	99	99	99	99999	999.99	99999
ac1														+	+	+	+	+	+	+	+	+	+	+	=				
ac2														+											+	=			
ac3														-											+	grw			
uc4																									+	prj	+		
+	+	A	3	2	3	3	2	2	2	3	1	1	2	3	2	1	1	2	3	2	3	1	2	2	3	3			
+	+	B	4	2	1	2	2	3	1	2	2	3	2	1	3	2	1	2	2	2	4	2	2	1	4	3			
+	+	C	3	1	2	1	2	2	1	1	2	2	1	2	3	3	3	2	3	3	3	2	1	2	4	3			
=	=	Total																											

"Compute clean compute clean" in sequence on the above table will produce the desired result. This technique can be used to deal with tables that have more than 25 columns or require more than 7 Calculation Strips.

CONSTANT ANNUITY LOAN REPAYMENT

	Annual Interest Rate	Period	Outstanding Balance	Interest Payment	Principal Payment	
ex	99.99	,,	999,999	999,999.99	99,999.99	999,999.99 ,
uc1		ftp		ftb		
uc2+		1				
jc3=		1		44000c		
uc4=	8.5a	12b				329.25d
uc5	+	/		*	#-	+=
uc6		stp		+	-	= s
+	8.50	1	44,000.00	311.67	17.58	
+	8.50	2	43,982.42	311.54	17.71	
+	8.50	3	43,964.71	311.42	17.83	
+	8.50	4	43,946.88	311.29	17.96	

- a) annual interest rate
- b) payment periods per year
- c) initial outstanding balance
- d) payment per period (interest + principal)

Notice how calculation "jc3" is used to establish the initial conditions for the first row. For other rows, "uc1" and "uc2" accomplish the task.

SUMMARY TABLES

SALES			
Month	Systems Sold	Manuals Only	Programs Only
ex	99,999	999,999	999,999
+	August 123	45	12
+	September 145	88	32
+	October 150	75	56
+	November 169	72	67
jc1	sts	stm	stp
=	Total 587	280	167
Summary Statistics			
ex	,,, ,,, ,,,	99,999.99	
uc1	fts ftm ftp		
jc2	+	+	=
jc3	+	+	/=
+	Manual Sales Per Program		1.15
jc2		/ +	%
+	Percent Of Manual Only Purchasers Buying The Program		59.64

In the above table, Store is used in the top table to save key numbers, so that they can be fetched and used to develop a few special summary statistics (the lower table).

A MORE COMPLEX CASH FLOW MODEL

	Jan	Feb	Mar	Apr
	999,999	999,999	999,999	999,999
ex				
zv				
jc1		fta	ftb	ftc
jc2	cum +	+ + +		
+ Bank Balance	1,540	2,142	2,727	3,592
* Monthly Interest Rate	1 1 1 1			
+ House Mortgage	513	513	513	513
jc1=		3		
jc2		+ prj + + +		
+ Other	800	824	849	874
= - Total Expenses	1,313	1,337	1,362	1,387
+ # Interest	15	21	27	36
+ Salary	1,900	1,900	2,100	2,100
+ Dividends			100	
= + Total Income	1,915	1,921	2,227	2,136
jc1		sta	stb	stc
= Net Income	602	584	866	749

Here is a trickier cash flow model that permits the compounding of interest on the cash balance. This table, however, requires one iteration of Compute for each column. In other words, the command to calculate this table correctly should be "compute c c c".

Notice also that "Other" expenses are now projected rather than entered as simple data. Additional Calculation Lines could be mixed into the table to meet various needs.

The example line could be changed ahead of the interest rate row to allow for greater precision in specifying the interest rate.

A FEW TRICKS

```

=====
Theatre Model
=====
Orchestra    Balcony    Floor    Total
=====
ex           999,999   999,999   999,999   999,999
rc5          +         +         +         =
zv
+ +  Rows           10        30        50
* *  Seats Per Row  30        35        40
*   Ticket Price    20        15        10
* *  Expected Occupancy Rate  95        75        55
=====
# +  Gross Revenues  5,700     11,813    11,000    28,513
=====
/  Seats Per Usher  30        40        60
jc1+          .499999   .499999   .499999
jc2          rnd       rnd       rnd
+ #  Ushers Required 10        20        19        49
cc
ex           .....      .....      .....      999,999
*  Cost Per Usher   25
=====
= -  Cost Of Ushers  1,225
-  Rental            5,000
-  Lighting          1,000
-  Cast              10,000
=====
=  Net Revenues     11,288
=====
```

This is a relatively simple table which starts out as four columns and ends up as one. Notice how the "rarely compute" option is used to avoid the calculation of meaningless numbers like a total "Ticket Price". This equation is cancelled when the table shifts to only one column.

Notice also how the "Ushers Required" is forced to be the first integer which is greater than the intermediate result dervied by its Calculation Strip.

A MAXIMUM, MINIMUM, AND AVERAGE OF A COLUMN

```
=====
      STAFF EMPLOYED
=====
ex          9,999,999 ,,,,
zv
jc1          sta  stb
uc2          +    fta * max sta
uc3          +    ftb * min stb
+
+   1975          12,890
+   1976          14,579
+   1977          15,678
+   1978          11,247
+   1979          10,391
+   1980          12,311
=====
avr  Average          12,849
=====
jc1          fta
=   Maximum          15,678
=====
jc1          ftb
=   Minimum          10,391
=====
```

Calculation Line 1 simply stores the value in the first row as the "maximum so far" and the "minimum so far." With each new row, the maximum and minimum is taken between the row's value and the stored values.

Finally, these "so far" values are fetched back. Notice that an "=" sign is used as the strip symbol for the last two rows. Since the values are fetched by the calculation lines above, it really does not matter what symbol is put here. However, using a terminator symbol insures that the "uc" Calculation Lines will not apply. Also, if other rows are later added to the table, no "live" strips are left hanging unterminated.

SOME BASIC STATISTICS

```

=====
          Centigrade                  Farenheit
=====
ex      99999.99 , , ,      99999.99 , , ,
uc1      ** =                   ** =
uc2      +                   * =
+       0.00                  32.00
+      100.00                 212.00
+      50.00                  122.00
+      15.00                  59.00
=====
          Average   Variance   Standard   Average   Variance   Standard
                           Deviation   Deviation
=====
ex      9999.99 , , 9999.99  9999.99  9999.99 , , 9999.99  9999.99 ,
jc1      ** --  **+  sqr    ** --  **+  sqr
avr +     41.25  1479.69   38.47   106.25  4794.19   69.24
=====
ex      " " " " " " " " " " " " " " " " 99999.999
jc1      +           * =           +=           +=
jc2      +           *=           *=
=====
=      Correlation           1.000
=====
```

The variance in the above table is based on a division by N rather than N - 1. More observations can be added simply by adding rows the upper part of the table.

STORING CONSTANTS FOR LATER USE

	Discount A	Discount B	Discount C	Discount D	
ex	
jc1=	10	20	35	50	
jc2	sta	stb	stc	std	
=	..				

Product		Discounts			Average
	None	A	B	C	Discount
ex	9999	9999	9999	9999	999.9
uc1	stq	str	sts	stt	stu
					=
uc2	+	+	+	+	
uc3	+fta*=	+ftb*=	+ftc*=	+ftd*=	
uc4	+	+	+	+	/=
uc19	ftq	ftr	fts	ftt	ftu
+	Program A	23	14	75	150
+	Program B	82	65	32	10
					300
					0
					39.0
					8.7

+	Computer A	23	10	5	0
+	Computer B	20	10	5	0
					0
					5.3
					5.7

The first portion of this table simply stores four constants. Notice how print design comment lines are used. When cleaned and printed, this first portion will not appear at all. In many cases, it is convenient to use such a technique to store constants, so they can be used later without having to use a Constant Calculation Line.

As far as the calculations are concerned, notice that the first Calculation Line simply stores the data on the line and that the Last Calculation Line fetches it all back. This means that calculations in the middle can use these columns for intermediate results without worrying about destroying the initial data. "uc3" does, in fact, replace the original data with an intermediate result.

SIMULATED TABLE LOOKUP

```

-----  

          Net Income Calculations  

-----  

ex           9,999,999  

+   Gross Income      87,601  

-   Expenses          47,590  

jc1          rnd sta  

=   Net Income       40,011  

-----  

          Tax Calculation Table  

-----  

          Bracket    Bracket    Bracket Marg.  

          Minimum   Maximum   Tax     Rate  

-----  

ex   9,999,999  9,999,999  9,999,999  999  , , , , , , , ,  

uc1=          1  

uc2          -  

uc3          -  

uc4min        * fta + ply  

uc5          * fta + nly  

uc6          1  

          fta + #  

* + =  

+   0          3,400      0          0  

+   3,400      5,500      0          14  

+   5,500      7,600      294        16  

+   7,600      11,900     630        18  

+   11,900     16,000     1,404      21  

+   16,000     20,200     2,265      24  

+   20,200     24,600     3,273      28  

+   24,600     29,900     4,505      32  

+   29,900     35,200     6,201      37  

+   35,200     45,800     8,162      43  

+   45,800     60,000     12,720     49  

+   60,000     9,999,999  19,678     50  

cc  

ex           9,999,999  , , , , , , , ,  

jc1          sta  

jc2          fta  

-----  

=+   Tax Due          10,231  

-   Prepaid Taxes      5,000  

=   Actual Tax Due     5,231  

-----
```

The interesting part of the above table is the tax calculation. The first four "usually compute" Calculation Lines will result in a one in the fifth column if the tax bracket is correct for the net income or a zero if the tax bracket is not correct. Calculation Lines 5 and 6 calculate the tax, however, the fifth column is included as a multiplier in Calculation Line 6, so the result will be zero unless the tax bracket is the correct one.

FITTING A POLYNOMIAL

```
=====
SOLUTION OF  $y = A * x^2 + B * x + C$ 
=====
Function Values
=====
      x          y
=====
ex      99,999,999.99
uc1      +*
,
---+ Value 1      10.00      123.00
+ Value 2      20.00      443.00
+ Value 3      30.00      963.00
=====
ex      ,,,      ,,,      ,,,      ,,,      ,,
jc1      /      +sty
jc2      /      -sta
=
jc1      +
jc2      +
jc3      +
jc4      +
+=
jc1      +
jc2      /
jc3      +
jc4      +
+=
jc1      +ftb*=      +fta*=      fty*=      ftb*=      ftc*=      stb*=      stc
jc2      -      +      -      -      -      -      =
=====
Coefficients Of The Solution
=====
      A          B          C
=====
ex      9999.99999   9999.999999   9999.999999
jc3      fta        ftb        ftc
=      1.00000     2.000000    3.000000
=====
```

This table is probably not for anyone who received less than a A- in algebra. The general algorithm solves the problem as you would do it by hand by reducing the number of variables and substituting from one equation into another.

The accuracy used in the example lines may be changed depending on the values for the function and the derived coefficients.

THE CLEAN COMMAND

The Clean command removes from the working area the example line, the zero values line, the column names line (used by the Bar command), and calculation lines and strips--preparing the table for final printing. (The .CLEAN print design command provides another, and perhaps more powerful, approach to cleaning--see later subsection.)

WHAT NEXT? CLEAN

Print design commands are unaffected by the Clean command.

AN EXAMPLE OF CLEAN

Before				After			
Profit Analysis				Profit Analysis			
Item	Cost	Sales	Profit	Item	Cost	Sales	Profit
.end				.end			
ex	999,999.99	999,999.99	999,999.99	Windows	125,167.88	242,234.58	117,066.70
aci	-	+	=	Mirrors	66,145.18	110,499.18	44,354.00
+	Windows	125,167.88	242,234.58	Storm Doors	48,091.23	57,120.11	9,028.88
+	Mirrors	66,145.18	110,499.18	.block			
+	Storm Doors	48,091.23	57,120.11	Total	239,404.13	409,853.71	170,449.58
.block				.end			
=	Total	239,404.13	409,853.71	170,449.58			
.end							

Tip: Design commands placed within a table tend to muddle its appearance. A good approach is to disguise design commands as ordinary lines of text by beginning the commands in position 8 instead of position 1. After Clean removes the first seven positions, the formerly-disguised commands become legitimate design commands.

OTHER USES

Clean may be used in other instances where the intention is to drop the first seven positions of each line and none of the lines begin with characters that might be confused with the specialized lines Clean eliminates. Consider the table below in which line numbers have been entered for sorting purposes.

ex		9999	9999
+	20	Item 1	10 23
+	21	Item 2	12 21
+	22	Item 3	15 19
	15		-----
=	10	Total	
	5		-----

For the above, the command "compute clean sort 1 7 clean" would produce-

Total	37	63

Item 1	10	23
Item 2	12	21
Item 3	15	19

The second use of the Clean command is simply to eliminate the line numbers used for the sort.

AN ODDITY

Clean never removes more than the first seven positions of a row even when a terminator symbol extends beyond position seven (e.g. "avr" with the "a" in position six). If such a situation is unavoidable, an easy solution is to start the text of the table in position 15 and use Clean twice in succession before printing.

ex		9999

+ +	Item 1	
+ +	Item 2	

avr	Average	

=	Total	

USE OF THE ".CLEAN" DESIGN COMMAND

In many cases, use of the ".clean" design command represents a more practical approach to cleaning a table. Consider the table below.

clean

	=====	1981	1982
	=====		
ex		999,999.99	999,999.99
	=====		
+	Item A	145.60	145.80
+	Item B	211.20	561.80
=	Total	356.80	707.60
	'-		

To view this table as it would appear cleaned or to print a cleaned version on paper, one need only type the command "print it" and answer the printer's prompt appropriately. This process will not effect the table itself. If the Clean command were used instead, the table in memory would no longer be useful for other than printing.

Note that subsequent design commands after the ".clean" may begin (if desired) in position 8 instead of position 1.

THE COMBINE COMMAND

The Combine command combines a table on disk with a table in the working area, generating a new table in the working area.

```
WHAT NEXT? COMBINE disk-file-name  
disk-file-symbols working-area-symbols
```

Following the Combine command, the user must specify three parameters. The first is the name of the disk-file with which the working file is to be combined. The second and third parameters are General Calculation symbols telling how the two tables should be combined. Each parameter is separated from each other by one or more spaces, and no spaces are permitted within a parameter.

Combine can be used in succession to add or perform a variety of calculations for a series of tables. Examples of the use of Combine are: summing the tables for each month to get a table for the year; taking a table of actual budget figures and subtracting it from another table of planned figures.

Like Compute, Combine also prints a warning message showing the number of "MINOR ERRORS."

THE FILES COMBINED

When Combine is used, the working area and the file on disk should both be tables that conform to the format requirements of the Compute command. Every row containing data in either table should have at least one valid symbol in the first seven positions of the line so that Combine can identify it as a row of values. This, again, corresponds to the rules for using Compute.

SPECIFYING THE GENERAL CALCULATION

Combine processes the table on disk and the table in the working area on a cell by cell basis. For example, row 5 column 6 of the table on disk will be combined with row 5 column 6 of the table in the working area to derive a new value for row 5 column 6 in the working area. This new value will replace the old one and will be presented according to the example lines in the working area. Text in the working area will be unchanged by Combine.

All corresponding cells are combined using the same formula. Thus, specifying how a calculation should be done for one cell is equivalent to specifying how the calculation should be done for all cells. The calculation symbols used can only be those of a General Calculation (see Compute). This calculation is specified in a special way as two parameters after the name of the disk-file to be combined.

The first one tells what is to be done with a cell from the table on disk. Normally, this parameter consists of a single mathematical operator.

The second parameter tells what is to be done with a cell from the working area. Normally, it consists of a mathematical operator and a terminator symbol specifying the result to be produced. For example, the command below adds a table on disk to a table in the working area.

```
combine sales.aug + ==
```

Notice that no blanks are permitted within a parameter. The command below recalculates the table in memory as a percent of a table on disk.

```
combine actual.82 / +%
```

Another way to visualize the meaning of these symbols is to think of an example line with two columns for which the initial value of the first column is the value of the table on disk and the initial value for the second column is the value of the table in memory.

	Disk Value	Working Area Value
ex	999999	9999999
combine table.1	??????	???????

The problem of specifying the calculation to Combine is a matter of identifying the appropriate symbols that would be placed on a single Calculation Line to achieve the correct result. Below, for example, the square of the first column is taken as a percentage of the second column. Again, notice that no spaces are left within the first or second set of symbols.

	Disk Value	Working Area Value
ex	999999	9999999
combine table.1	+**+	/%

Tip: A formula for Combine can be tested on particular values before using it by transforming the solution into a Calculation Line and using the Compute command. For the previous formula, the test table below would suffice.

	Disk Value	Working Area Value
ex	999999	9999999
aci	+**+	/%
+	3	12

NONCONGRUENT TABLES

Ordinarily, Combine is used on tables having the same numbers of rows and columns. If, however, one table has fewer columns or fewer rows, values will be combined only where corresponding cells exist. Where the calculation is not applied, the working area's value will remain unchanged.

A warning message is generated if the number of rows in the two tables are not the same.

MERGING TABLES

Combine can also be used somewhat deceptively to merge two tables rather than to do a proper arithmetic operation between them. Suppose, for example that a new example line were added temporarily to the table below as shown. A Combine with another table could be used below to enter values for the new columns.

	1979	1980		
ex	9999.99	9999.99		
ex			9999.99	9999.99
*	Item A	14.51	18.32	
*	Item B	15.21	16.23	
=	Total	29.72	34.55	

After having done this, the example line could be modified to show four columns of numbers, and headings, calculations lines, etc. could also be added.

EXAMPLES

Working Area:		Disk File: ESTIMATES			
Expenses To Date		Estimates For 1982			
ex	999,999	ex	999,999		
+	Rent	12,567	+	Rent	18,177
+	Salaries	44,678	+	Salaries	62,356
+	Supplies	21,441	+	Supplies	25,144
+	Other	11,233	+	Other	13,490
=	Total	89,919	=	Total	119,167

Result of: combine estimates / +%

Working Area:		
Expenses To Date		
ex	999,999	
+	Rent	69
+	Salaries	72
+	Supplies	85
+	Other	83
=	Total	75

Note: The example line of the table in the working area could have been changed prior to the use of Combine to have more precision in the results.

Working Area:		Disk File: JAN		Disk File: FEB		Disk File: MAR	
		Item 1		Item 1		Item 1	
ex	9999999	ex	9999999	ex	9999999	ex	9999999
+	Store A	+	Store A 1	+	Store A 3	+	Store A 2
+	Store B	+	Store B 3	+	Store B 3	+	Store B 1
=	Total	=	Total 4	=	Total 6	=	Total 3

Result of: combine jan + = combine feb + += combine mar + ==

Working Area:

Item 1	
<hr/>	
ex	9999999
+	Store A 6
+	Store B 7
=	Total 13
<hr/>	

Note: The first use of Combine in the series simply sets the values of the table in the working area to those of the disk file "JAN".

Working Area:

Disk File: PRICES

Disk File: ORDERS

Dollar Value Of Orders			Prices			Orders						
	Price 1	Price 2	Total		Price 1	Price 2		Price 1	Price 2	Total		
ex	999,999.99	999,999.99	999,999.99	ex	999.99	999.99	ex	999999	999999	999999		
ac1	+	+	=	+	Item 1	100.00	80.00	ac1	+	=		
+	Item 1			+	Item 2	200.00	160.00	+	Item 1	45	22	67
+	Item 2							+	Item 2	11	33	44
=	Total							=	Total	56	55	111

Result of combine prices + = combine orders + *= compute

Dollar Value Of Orders				
	Price 1	Price 2	Total	
ex	999,999.99	999,999.99	999,999.99	
ac1	+	+	=	
+	Item 1	4,500.00	1,760.00	6,260.00
+	Item 2	2,200.00	5,280.00	7,480.00
=	Total	6,700.00	7,040.00	13,740.00

Note: The correct dollar totals are calculated by the final use of Compute on the result produced by the two uses of Combine.

GRAPHICS

T/Maker currently has a single graphics command to produce bar charts of various types. These may be displayed on the screen or printed without the necessity of graphics hardware. Where the hardware has the capability to produce special characters or colors, T/Maker can usually be customized to make use of such features (see T/Modify Option 3 in the Customization Notes).

Charts can be developed with a minimum of effort. Often, it is only necessary to identify the name of a row or column to produce a bar chart.

TABLE OF CONTENTS

THE BAR COMMAND	GR-1
DESIGNATING THE NUMBERS TO BE CHARTED	GR-1
THE "IT" VARIABLE	GR-2
CONTROL OPTIONS	GR-2
BAR NAMES	GR-5
INTERPRETING THE BARS	GR-6
SOME NOTES ON USING BAR	GR-7
PRESENTATION OPTIONS	GR-8
DISPOSITION AND OTHER OPTIONS	GR-9
INTERSPERSING BARS	GR-11
STACKING BARS	GR-13
A COMPLEX EXAMPLE	GR-15
LIMITS AND ERRORS	GR-17

THE BAR COMMAND

The Bar command prepares a bar chart based upon the values in either one column or one row of a table. This table must be the working file and must conform to the format requirements of the Compute command.

Charts constructed with Bar can either be displayed on the screen or entered into the working file for editing or printing.

WHAT NEXT? **BAR { row-or-column-name or IT } options END**

The row or column name is just a character string (of less than 30 characters) which is specified in the same way as strings used with the Drop, Keep, Find, and Replace commands.* The Bar command looks for the string specified in the working area. Depending on where it finds the string, it charts a corresponding row or column of numbers.

If the word "it" is given instead of a string or name, then the row of numbers which is the top line of the screen or the top line of the frame (when in Frame Mode) is charted.

A variety of options can be used to determine which values are to be charted (control options), to determine how the bar chart is presented (presentation options), and to determine the disposition of the bar chart (disposition options).

All options can be specified in response to a general prompt for options, or they may be specified in advance by placing them after the row or column name.

The word "end" is typed to exit the Bar command and return to the Universal Prompt.

DESIGNATING THE NUMBERS TO BE CHARTED

In order to make a bar chart, a row or column of numbers must be designated. In most cases, it is sufficient to provide the name appearing at the top of a column or the name at the side of a row. This name (or character string) should exist in the table in only one location. The Bar command begins looking for the string

* The string must be enclosed in single or double quotes if it contains a blank or begins with a single or double quote. The enclosing quotes should themselves not appear in the string. (See the Find command in List Processing for additional details.)

starting with the top line on the screen. If it can not find the name, an error message is produced.

	Store A	Store B	Store C	TOTAL
ex	999,999	999,999	999,999	999,999
aci	+ +	+ +	+ =	
+	ITEM 1 10	11	12	33
+	ITEM 2 14	15	16	45
+	ITEM 3 20	22	33	75
=	Total 44	48	61	153

For the above table, the columns could be charted by the commands below.

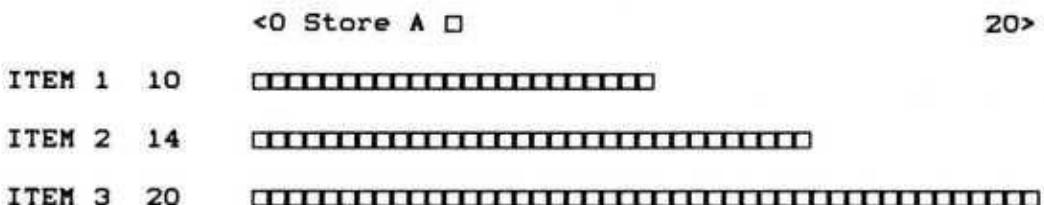
BAR 'Store A' BAR 'Store B' BAR 'Store C' BAR TOTAL

Rows could be charted by the following commands.

BAR 'ITEM 1' BAR 'ITEM 2' BAR 'ITEM 3' BAR Total

Notice in the table that the "TOTAL" column has been spelled with all capitals to avoid confusion with the "Total" row.

The command "BAR 'Store A'" on the above table would produce a chart similar to this:



Notice that all the numbers in the column 'Store A' have been included in the chart except for the total. When a column of numbers is charted, Bar typically includes only rows which have a plus sign in the first position. (The "ALL" option described later can be used to override this rule.)

Note: It is usually a simple matter to prepare a table that is suitable for both the Compute command and the Bar command. If worst comes to worst, the first calculation strip of the table can be randomly spotted with plus signs to show which rows should be charted. This calculation strip need never be ended by a terminator symbol. Thus, Compute would do a senseless but harmless addition of rows.

When a row of numbers is charted, all columns are included except those for which the model number on the example line begins with a zero (i.e. "0"). Beginning a model number of an example line with a "0" has no effect on the Compute or Combine commands. In the table used previously, changing the example line to the one below would suppress displaying the "TOTAL" column whenever a row was charted.

	Store A	Store B	Store C	TOTAL
ex	999,999	999,999	999,999	099,999

THE IT VARIABLE

Instead of specifying a name to designate the row or column to be charted, the word "it" may be used. In this case, the row at the top of the screen (or the top of the frame, when in Frame Mode) will be charted.

Tip: While working in the editor with the cursor on a row of interest, one can simply quit the editor, give the command "bar it", and the result will be a chart of that row. It is often faster to examine numbers in this way than it is to move the screen around on a wide table.

CONTROL OPTIONS

Control options are used to define the numbers which are to be charted. Generally, the Bar command decides whether a row or column of numbers is charted based on where it finds the name specified. In some instances, it may make the wrong decision. The decision can be overridden by using one of the two following control options. (These options are most frequently used when the string given as the "name" is really just one of the numbers in the table.)

row This option forces the Bar command to chart the row of values from the line on which the name was located.

Example: bar 33 row

column This option forces the Bar command to chart the column of values corresponding to the position in which the name was located. The word "column" can be abbreviated to "col".

Example: bar 33 col

As noted previously, the columns of numbers included in the chart of a row can be controlled with the first position of the model number for the column (a zero is used to suppress charting a particular column). The rows used when a column is charted can also be controlled by the presence or absence of a plus sign in the first position of the row. The options below can be used to override this logic.

all If a column of numbers is being charted, the "all" option causes all rows containing numbers to be included even though they do not necessarily contain a '+' sign in position 1. If a row of numbers is being charted, the "all" option causes all column values to be included even though the model number on the example line corresponding to some of the columns may start with a "0".

Example: bar 'Store A' column all

some The "some" option overrides the "all" option and causes the Bar command to apply its normal rules for deciding which values to include in the chart.

The variable being charted may be changed by using the following option.

bar Giving the word "bar" followed by a new row or column name causes the Bar command to redo the chart with the new values. When this is done, some presentation options of the first chart may be kept for the next. Normally, this option is used when a series of charts are to be made all using, say, the same maximum and minimum. If instead, the command "end bar" is used, then the current use of the command would be ended, and the next use would start over from scratch.

Example: bar 'Store A' insert bar 'Store B' insert

BAR NAMES

When creating a bar chart it is helpful to have names appearing next to the bars. When a column of numbers is charted, Bar will take the name of each row from position 8 to the start of the first model number on the example line. The following diagram shows the assumption.

```
-----  
ex      999, 999  999, 999  
+ This-is-where-the-name-is-assumed-to-be    20   21
```

The space available for storing names is limited but should be sufficient for most charts. If the error message "NAMES TOO LONG" occurs, a dummy model number can be put on the example line temporarily so that less space will be used.

ex	,	999,999	999,999
+	This-is-less-space	20	21

When charting a row of numbers, Bar makes no assumptions about the names of columns. Therefore, a special line can be included in a table to assign names to columns. This line begins with the abbreviation "CN" for "Column Names". It is typically put just under the example line with the names located spacially under the model numbers to which they apply. The example below shows the correct technique.

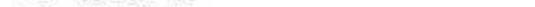
	GROSS	COSTS	NET	%MARGIN
--	--------------	--------------	------------	----------------

The Compute and Combine commands will ignore such a line, and the Clean command will remove it; therefore, a "column names" line can be made a standard part of all tables. The names on the line can also be used for the purpose of identifying a column to be charted.

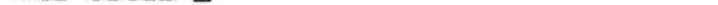
BAR X MARCH 2011

INTERPRETING THE BARS

Assuming a column names line had been included in the initial example table, the instruction "bar Total" would produce a chart similar to this:

	<0 Total □	61>
Store A	44	
Store B	48	
Store C	61	

The line at the top of the bar chart shows (space permitting) the range of the bars, the variable being charted, and the character used to draw the bars. All these parameters can be controlled by options. For example, the chart could be redone with a different maximum and minimum as shown next.

	<48 Total	□	60>
Store A	44	<	
Store B	48		
Store C	61		>

Now the range of the bars goes from 48 to 60. Notice how the Bar command presents values which are less than the minimum (Store A), equal to the minimum (Store B), and greater than the maximum (Store C).

The Bar command draws bars somewhat differently depending on whether the maximum and minimum are both greater than or equal to zero, the maximum and minimum are both less than or equal to zero, or the maximum is greater than zero and the minimum is less than zero. The previous graph shows the first case; the other two cases are shown next. These charts were made by changing the numbers in the example table.

<-61 Total □

0>

Store A -44



Store B -48



Store C -61



In the case of all negative values, the bars are drawn in the opposite direction so that all finish at a common spot.

<-44.576923076923 Total □

61>

Store A -44



Store B 48



Store C 61



In the case of positive and negative values, the bars are drawn out from a center point depending on whether the number is negative or positive. In such instances, the Bar command must adjust the minimum value so that the zero point will fall exactly between two positions.

SOME NOTES ON USING BAR

When using Bar in an interactive fashion, the first thing to do is to get a copy of the appropriate table as the working area. It is wise to rename the table at this time if the table is to be replaced by charts created during the session.

Next, the Bar command can be given as a response to the Universal Prompt with the row or column name and whatever options are known in advance.

BAR Total ALL

Then, a carriage return is touched. At this point, a bar chart will be produced on the screen. Other options can then be specified, and the chart can be redrawn with the new options simply by touching a carriage return. When finished, "end" is typed to return to the Universal Prompt.

Tip: To make a fresh start on another chart from the same table, specify "end bar" instead of just "end" as the terminating command. A prompt will appear for the row or column name without a delay which would occur if only "end" were typed.

The Bar command can also be used as a part of a prespecified series of commands. The resulting charts can be printed or saved on disk. Examples of using Bar in this fashion are given in later sections.

PRESENTATION OPTIONS

The options listed in this section can be changed as often as desired so long as "end" has not been typed to signal the end of the list. These options are used to control how the chart looks on the screen or on paper. In some cases, a chart which is wider than the screen may be created. When this happens, the chart should be placed in working file and the editor can be used to examine it. For charts which are longer than the screen, the "long" option can force the entire chart to be displayed.

width number Changes the width of the bars to the number of lines specified. The values may be between 1 and 100. If no specification is given or if the number specified is out of the working range, Bar will pick a suitable number itself.

Example: width 5

space number Changes the spacing between bars to the number of lines specified. The value may be between 0 and 100. If no specification is given or if the number specified is out of the working range, Bar will pick a suitable number itself.

Example: space 1

length number Changes the length of the bars to the number of positions specified. The value may be between 10 and 200. If no specification is given or if the number specified is out of the working range, Bar will pick a suitable number itself. "Length" may be abbreviated to "len".

Example: len 50

names number Sets the number of positions used to show the names of the bars to the number given. The value should be between 0 and 100. If no specification is given or if the number specified is out of the working range, Bar will pick a suitable number itself. If a value of zero is given, the names will not be displayed as a part of the chart.

Example: names 10

values number Sets the number of positions used to show the values corresponding to the bars drawn. The number should be between 0 and 100. If no specification is given or if the number specified is out of the working range, Bar will pick a suitable number itself. If a value of zero is given, the names will not be displayed as a part of the chart. Normally this option is only used when two or more charts are to be put together; its purpose is to insure that the same amount of space is used in both charts for displaying values, so the bars will start at the same position.

Example: values 10

numbers	Causes the lines in the bar chart to be numbered so that two or more different bar charts could be sorted together to form a single chart with interspersed bars.
nonumbers	Reverses the above.
character char	Changes the character used to draw the bars to the character given. Normally, reverse video is used if it has been specified and the chart is only to be placed on the screen. Otherwise the character used to draw the frame when the Editor is in frame mode is used. "Character" may be abbreviated to "char". A printer translation table can be used to create special graphics characters on many printers (See T/Modify Option 3 of the Customization Notes.)
	Example: char #
maximum number	Changes the maximum number used as the upper end of scale to the number given. It should be greater than the minimum. "Maximum" may be abbreviated to "max".
	Example: max 456.78
minimum number	Changes the minimum number used as the lower end of the scale to the number given. It should be less than the maximum. "Minimum" may be abbreviated to "min". If the minimum is less than zero and the maximum is greater than zero, Bar may be forced to adjust the minimum to a particular number so that a zero falls exactly on a spot between two positions. This adjustment can be avoided by selecting a "minimum", "maximum", and "length" which achieve this objective.
	Example: min -20 max 60 length 80
long	Causes all bars in the bar chart to be scrolled on the screen. The listing may be interrupted and restarted by touching the space bar. When this option is in effect, a key must be touched at the end of the display to return to the next prompt.
short	Causes only enough bars to be put on the screen so that no scrolling occurs. This option is in effect by default.
nobars	Suppressing putting in the bars into the bar chart. This option is rarely used, but one example of it is shown later.
bars	Overrides the above, and causes the bars to be displayed.

DISPOSITION AND OTHER OPTIONS

Disposition options control what happens to the current chart. When working on the screen, no disposition options are normally used and the session is simply terminated by the word "end". If the current chart is to be kept for printing or whatever, the "replace" option can be used.

replace Causes the working file to be replaced by the current bar chart. Normally, an "end" would be typed immediately afterwards although one could still change some of the presentation options and type "replace" again. It would be too late, however, to look for a new variable or to change some control options, since the working file which held the data would now be destroyed.

In some cases it is desirable to accumulate charts into the working file without destroying the file itself. The following option represents a way to do this.

insert Causes the current chart to be inserted into the working file ahead of the first line in the working file. The line numbers are altered at the same time so that the top line on the screen remains the same as it was before the insert took place. When an insert operation is performed, the various presentation options are fixed so that any following charts will be done using the same options unless explicitly overridden by the user.

In order to do anything more complicated than what has been described so far, the Bar command should always be given with the top line on the screen being the first line in the working file.

When Bar is entered in this fashion, it can distinguish between the charts which have been inserted into the file and the original working file. Since an Insert operation causes the line number of the top line to be increased by the length of the chart inserted, the Bar command always assumes that charts are stored before this line and that the initial working file falls after it.

The options below are based on this assumption and will not work otherwise.

file The initial working file is redrawn on the screen. This is useful when the chart of a previous variable is on the screen and one needs help remembering the name of the next variable to chart.

chart The chart or charts accumulated ahead of the working file are displayed.

restart The chart or charts accumulated ahead of the working file are destroyed. Sometimes it is useful to create a chart and insert it only for the sake of fixing a suitable maximum for the charts to follow. In this case, "restart" can be used to wipe out such a chart while continuing with the various options it fixed in its wake.

cut This option destroys the initial working file and adjusts the top line on the screen to be the first line in the file. Presumably, these lines correspond to the chart or charts accumulated.

The next option is used to create charts in which each bar is made up of pieces of other bars. This can only be done when all the numbers in question are positive.

stack The bars from the current chart are stacked onto the end of the bars of the chart previously inserted.

INTERSPERSING BARS

Bar charts can be developed with the bars from two or more charts interspersed. Consider the table below.

Sales Figures				
	Month	Retail	Dealer	Total
ex		999,999	999,999	999,999
acl		+ + =		
+	January	49,143	38,156	87,299
+	February	47,423	47,943	95,366
+	March	52,010	56,145	108,155
+	April	56,966	64,157	121,123
=	Year To Date	205,542	206,401	411,943

Suppose one wanted to have a bar chart in which the bars of the Retail Sales and the Dealer Sales are interspersed. The most simple approach is the following. First, give the Bar command with the first line of the file at the top of the screen and bar the variable that has the values that occupy the most space. This will save having to specify the "values" option to insure that all charts are aligned with each other. Presumably all variables have the same row or column names, so the "names" option would not be required to insure alignment.

Next, the various presentation options including a suitable maximum and minimum should be given. Usually, variables in an interspersed chart use the same scale and presentation options. Then, the numbers option is specified so that the bars will be numbered in a manner suitable for sorting. For each chart an appropriate character can be selected using the the "character" option.

When the first variable looks properly barred, the "insert" option is used to insert the chart into the file. "File" can then

be specified to see the original table again. The next variable can be charted by specifying "bar" and the next row or column name. Presumably, a different character is used for the bars of this chart. When the second chart is correct, it can be "inserted."

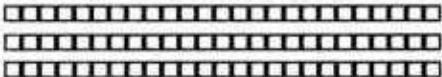
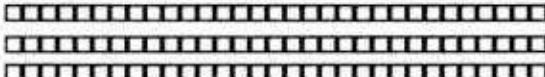
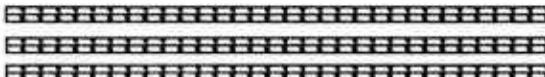
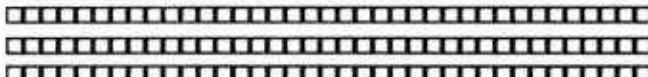
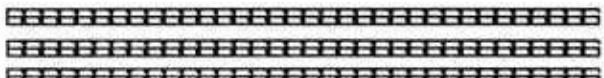
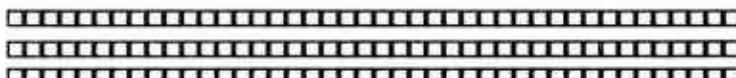
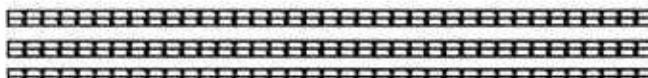
One can continue in this manner to insert the charts of other variables (space permitting). In order to eliminate the original working file, the "cut" option is used and followed immediately by the "end" command to return to the Universal Prompt.

Next, the Sort command is needed to sort the charts into order. The command "sort 1 3" will work, since the numbers placed into the file by the numbers option are in positions 1 to 3. Finally, the Arrange command is used, "arrange 6 300 end", to eliminate the five positions which contain the numbers and two separating spaces.

Putting all this together produces a command sequence like that shown. If this command were placed on a single line ahead of the table and executed with the Do command, it would produce the chart shown. This command would work independently of the number of months being barred.

When preparing a complex chart, it is often easier to try to specify the entire command in advance and then use the Do command; this means one does not have to retype the entire command again if things go awry.

```
BAR Retail MAX 75000 VALUES 9 CHAR E NUMBERS INSERT
BAR Dealer CHAR D INSERT CUT END
SORT 1 3
ARRANGE 6 300 END
```

	<0 Dealer □	75000>
	<0 Retail □	75000>
January	38,156	
January	49,143	
February	47,943	
February	47,423	
March	56,145	
March	52,010	
April	64,157	
April	56,966	

STACKING BARS

Drawing on the example of the previous section, suppose a bar chart were desired in which the bars of the Retail Sales are to be stacked on top of the bars of Dealer Sales. The first problem is to pick an appropriate minimum and maximum; this is easiest to do if there is a variable which is the total of all the variables to be stacked. In this case, one could begin by giving the Bar command and charting the Total column. The presentation options could then be set to achieve a chart with the proper appearance.

Next, the "nobars" option is used and the result could be inserted into the file. This only serves to enter the names and the values for the Totals. As a convenience, the "nobars" option is turned off automatically when an insert is done. Now, one can type "bar Dealer" and use the "character" option to pick an

appropriate character. Then, the "stack" option is used to stack the Dealer bars.

Then, one would "bar Retail", change the character to something other than the one used for "Dealer", and again stack the bars. Finally, "cut" and "end" are used as when creating a chart with interspersed bars.

In total, the command would look something like the following.

```
BAR Total NOBARS INSERT  
BAR Dealer CHAR ■ STACK  
BAR Retail CHAR □ STACK CUT END
```

If this sequence were placed on a single line above the table and executed with the Do command. It would produce the following chart. When doing such charts, it is worth remembering that because of errors introduced by rounding, the length of a bar made up of a number of pieces may be slightly longer or shorter than it would have been had the bar length been calculated directly from the total of contributing values.



A COMPLEX EXAMPLE

The command below mixes interspersed and stacked bars. If this command were placed on a single line above the table and executed with the Do command, the result on the next page would be achieved. Notice also that the Bar command is ended after the chart for Company B, so as to use a different maximum for Company A.

```
BAR 'B Total' NUMBERS VALUES 3 WIDTH 3 SPACE 2 NOBARS INSERT
BAR 'B Passed' CHAR █ STACK BAR 'B Failed' CHAR □ STACK END
BAR 'A Total' NOBARS NUMBERS WIDTH 3 SPACE 1 INSERT
BAR 'A Passed' CHAR ■ STACK BAR 'A Failed' CHAR □ STACK CUT END
SORT 1 3 ARRANGE 6 300 END
```

Quality Control Results

	Company A			Company B		
	Passed	Failed	Total	Passed	Failed	Total
ex	99999999	99999999	9999999	99999999	99999999	99999999
cn	A Passed	A Failed	A Total	B Passed	B Failed	B Total
aci	+	+	=	+	+	=
+	1975	113	32	145	45	5
+	1976	120	45	165	49	7
+	1977	130	67	197	54	13
+	1978	138	75	213	73	14
+	1979	142	83	225	91	12
						103

<O A Passed ■		225>
<O A Failed □		225>
<O B Passed ■		103>
<O B Failed □		103>
1975 145		
1975 50		
1976 165		
1976 56		
1977 197		
1977 67		
1978 213		
1978 87		
1979 225		
1979 103		

LIMITS AND ERRORS

The bar chart including names, titles, bars, etc. may not be longer than then the maximum line length accepted by the Editor (i.e. 300 characters). The chart may not consist of more than 100 bars. If an option is misspelled, a beep will normally be sounded, and, of course, the chart will not be modified in the way anticipated. Aside from the fatal error messages particular to Bar which are shown below, the general error message "OUT OF SPACE" may occur if the working file including the charts inserted exceeds the usual space restrictions.

string NOT FOUND

The string given as the row or column name was not located in the file anywhere after or including the top line on the screen.

TOO MANY VALUES

An attempt was made to develop a chart with more than 100 values. The only way around this problem is to cut the data into two pieces, and bar each piece separately.

VALUES TOO LONG

The space reserved for saving the character representations of the values is limited, so lots of long values (10 or more positions per value) may cause this message. This problem can be avoided by shortening the model of the number on the example line perhaps losing some of the less significant digits or by cutting the data into pieces.

NAMES TOO LONG

The space reserved for saving names is limited, so lots of long names (20 or more positions per value) may cause this message. The data can be cut into pieces or the names can be shortened (see "BAR NAMES").

NO EXAMPLE LINE or NO MODEL NUMBERS ON THE EXAMPLE LINE

The Bar command could not find a suitable example line with which to interpret the values. Check to see that an example line with some model numbers on it is present ahead of the values to be charted.

CAN NOT STACK NEGATIVE NUMBERS

An attempt was made to use the stack option with negative numbers or with a minimum that was negative. This would not produce a meaningful chart, so an error message is generated.

LIST PROCESSING

The eight commands described in this section are often used to process lists. For example, the Keep and Drop commands select lines from a list, the Sort command sorts the elements of a list into alphabetical or numerical order, and the Tally command counts the frequency of values in a list.

But these "list processing" commands can also be used for a variety of word processing and data processing applications as well. Sort, for example, can be used to sort the rows in a table. The Find and Replace commands are useful in many contexts besides list processing. (They are described in this section merely because they make use of a "string" parameter, a feature that is shared with other list processing commands.) The Match command, too, is useful for highlighting differences between two files whether they contain lists or not.

TABLE OF CONTENTS

THE ARRANGE COMMAND	LP-1
THE FIND COMMAND	LP-2
THE REPLACE COMMAND	LP-2
THE DROP COMMAND	LP-3
THE KEEP COMMAND	LP-3
THE MATCH COMMAND	LP-4
THE SORT COMMAND	LP-6
THE TALLY COMMAND	LP-11

THE ARRANGE COMMAND

The Arrange command reorders the sequence of columns in the working area.

WHAT NEXT? ARRANGE left-position right-position, etc. END

The columns to be moved are specified by numbers following the command name. The first number marks the left boundary of the column to be moved, and the second number marks the right boundary of that column. One or more columns can be specified for moving, each receiving its own number-pair. The list of number-pairs must be terminated by the word "end".

The new order of columns is determined by the sequence of number-pairs following the Arrange command-name. In the example command shown below, the number-pair 25-26 defines a column that will now become the first column, occupying positions 1-2. The next number-pair, 10-23, defines a column that will become the second column in the new order. Columns that are not described by a number-pair will be lost.

All lines in the working area except print design commands will be reordered.

AN EXAMPLE OF ARRANGE

ARRANGE 25 26 10 23 END

BEFORE							AFTER						
1	5	10	15	20	25	30	1	5	10	15	20	25	30
.....
Male	Single	17			17	Single							
Male	Married	42			42	Married							
Male	Married	51			51	Married							
Female	Single	28			28	Single							

Note: Columns of blanks may be moved, too, in order to provide white space between columns.

THE FIND COMMAND

The Find command advances the top line of the screen (or of the frame, when in Frame Mode) to the first line in the working area that contains the specified string.* If the string is not found, the top line is not advanced (and a beep may sound).

WHAT NEXT? FIND string

(The Find command, which is used outside the editor, is the counterpart of the "Search For A String" keystrokes, used when one is inside the editor.)

THE REPLACE COMMAND

The Replace command replaces all occurrences of a string with another string. The Replace command-name is followed, respectively, by the string to be replaced and the new string replacing it.

WHAT NEXT? REPLACE old-string new-string

(The Replace command, which is used outside the editor, is the counterpart of the "Replace Strings" keystrokes, used when one is inside the editor.)

THE DROP COMMAND

The Drop command deletes all lines that contain the specified string from the working area.

WHAT NEXT? **DROP** string

THE KEEP COMMAND

The Keep command keeps all lines that contain the specified string in the working area. All other lines in the working area are discarded.

WHAT NEXT? **KEEP** string

*HOW TO SPECIFY A STRING?

The Find, Replace, Keep, and Drop commands involve the specification of strings. A string is any series of characters. However, a few rules must be obeyed when specifying a string.

If a string contains no blanks and does not begin with a single or double quotation mark, it can be specified just by typing it as is.

Examples: alphabet 4,5 Jane's &#!\$\$.##'''

But if a string contains a blank or starts with a single or double quotation mark, the whole string must be enclosed in quotation marks. Also, the enclosing quotation marks (single or double) must not be present anywhere within the string.

Examples: 'NY NY' 'He said,' "'s"

Strings must be specified exactly (letter for letter, blank for blank, upper or lower case) as they occur in the working area. This means, incidentally, that a string containing a blank--for example, "the assignment"--will not be found if it appears in the text with more than one blank between the two words or if the two words fall on different lines.

THE MATCH COMMAND

The Match command contrasts the working area with a file on disk. It generates a report of the differences between the two files. The report becomes the working area.

WHAT NEXT? MATCH disk-file-name

The name of the file on disk is the only additional parameter the user provides.

THE MATCH REPORT

The Match report lists lines that do not have exact matches. In this report, the label "W>" denotes a line in the the working area that is not identically present on the disk file, and the label "D>" denotes a line on the disk file that is not identically present in the working area. The line numbers of such unmatched lines are also reported.

The Match report becomes the working area, replacing the previous working area. Thus, if you want to preserve the original working file it is recommended that you rename the working file before using this command. If this recommendation is followed, a Save command invoked after the appearance of the Match report will not destroy the original working file.

The report can be easily sorted (using the Sort command) by line number or file of origin (i.e., "D>" or "W>") or both. Such sorting may make the report more readable.

The Match command ignores blank lines throughout and no attempt is made to match them. When the disk file or the working area contains multiple instances of lines shared by both files, the Match command may confuse the correspondence between lines. Usually, it is obvious when this problem has occurred.

Tip: If the report produced by Match appears incorrect, it is often possible to produce a different result by reversing the roles of the two files (i.e. make the working area the file on disk, and make the file on disk the working area). Then try Match again.

AN EXAMPLE OF MATCH

The Disk File		The Working Area		The New Working Area			
Table Of Weights		Table Of Weights		S D) John	189	180	
	1980	1981		7 W) Alice	8	25	
John	189	188	Mary	115	120		
Mary	115	120	Bob	157	158		
Bob	157	158	Alice	8	25		

Tip: One interesting use of Match is in conjunction with the Keep and Drop commands. Consider the list below:

<u>Employee Number</u>	<u>Job Classification</u>
000001	Tech01
000002	Tech01
000003	Admin01
000004	Admin02
000005	Admin03
000006	Tech02
000007	Admin01

Suppose you wanted a list of everyone classified as "Tech01" or "Admin02." One solution is to use the following commands.

- 1) "DROP Tech01"
- 2) "DROP Admin02"
- 3) The list will now consist of the complement of the the list that is actually desired. Then, by MATCHing the current list against the original list on disk, the correct list will be achieved.

THE SORT COMMAND

The Sort command organizes the elements of a list into alphabetical or numerical order. This command sorts lines based on "key zones" that are identified by left-position and right-position parameters you provide following the command-name. Only lines in the working area are sorted.

WHAT NEXT? **SORT** { options, if any } left-position right-position

Three options may be specified. One permits you to specify whether the sort should be done in ascending or descending order; another permits you to specify an alphabetic or numerical sort; and the third permits you to specify the set of lines to which the sort should apply. Whenever options are used, they must be given immediately after the word "sort" and before the "left-position" specification. Each of these options is described in more detail below. (Specifying no options defaults to a sort in ascending order done upon all lines in the working area.)

THE SORT KEY ZONE

Whenever a sort is to be performed, it must be done relative to a particular zone of each line. This zone is called the "Key" or "Key Zone." In the list below, for example, we might want to sort the elements by "Age," "Sex," or alphabetically by "Name". The column positions associated with the desired sorting characteristic must be specified. The key cannot exceed 50 positions.

1	5	10	15	20	25
.....					
Name		Age		Sex	
John		25		M	
Mary		33		F	
Margret		15		F	
Tom		42		M	
Barbara		25		F	

The key merely consists of the left position and the right position of the desired column. For example, "Sex" begins in position 22 and ends in position 22. Thus, the following specification would be given in order to sort the list by sex.

SORT 22 22

The Info command (outside the editor) or the Information keystrokes (inside the editor) can be used to put a column position ruler at the top of the screen, making it easier to identify the starting and ending positions key zones. Additional blanks on either side of a key zone will not pose a problem, so the command to sort by sex might just as well have been entered as:

SORT 20 25

OPTIONS

Ascending or Descending Order

Ascending or descending order is specified by typing the letter "a" for ascending order or the letter "d" for descending order.

**A ASCENDING ORDER
D DESCENDING ORDER**

Default is ascending order.

For example, to sort the previous list in descending order by name (i.e., z to a), the correct command would be:

SORT D 1 10

Alphabetic or Numeric Values

A second optional letter specifies whether the values in the zone are numbers or alphabetic characters:

**C (ALPHABETIC) CHARACTERS
N NUMBERS**

Default is "c".

To sort the example list by age in ascending order, the correct command is:

SORT N 15 16

or

SORT A N 15 16

Note: When a character sort is performed, two key zones are compared one character at a time proceeding from left to right. A blank is the lowest character, thus " X" (i.e., a space followed by "X") is less than "AX". When a numeric sort is performed, all characters in the key are converted to a single number which is then used as the basis for comparisons. Thus, "-3" is less than " 3" in a numeric sort, but larger than it in a character sort. When dealing with all positive numbers aligned with each other, either type of sort will work.

Choosing Lines to be Sorted

The third option allows you to specify the lines to which the sort should apply. Three choices are provided:

ALL Sorts all lines in the working area.

P for "partial"--this sorts within blocks of consecutive lines that have any non-blank character in the sort key zone (note: this option will not sort across a blank line or a line with all blanks in the sort key zone).

S for "some"--sorts within blocks of consecutive lines that have a plus sign in the first position of the line (note: this option will not sort across any line that does not have a plus sign in the first position).

The "partial" option is handy for making sure that the list title is not sorted along with list elements. In the example list below, a "partial" sort will sort just the list elements because the title line is separated from lines of elements by a blank line (a line with all blanks in the sort key zone). This line is therefore a barrier across which lines will not be sorted. Only the lines in bold, below, would have been sorted by a partial sort.

Name	Age	Sex
JOHN	25	M
MARY	33	F
MARGRET	15	F
TOM	42	M
BARBARA	25	F

Keeping a few extra blank lines in a list often means it can be correctly sorted by a partial sort without having to worry about reordering headings or other text information.

A "some" sort is often useful for tables. It sorts data lines within groups but not across them. In the table below, only the bolded lines would be sorted. Lines without plus signs in the first position form barriers to the sort.

Item	Amount
ex	9,999.99
+ MORTGAGE	500.00
+ CAR PAYMENT	275.00
+ OTHER	550.00
= - TOTAL EXPENSES	1,325.00
+ SALARY	1,480.00
+ CONSULTING	300.00
= + TOTAL INCOME	1,780.00
= NET INCOME	455.00

In many lists or tables where sorting is desired, the first position of each line can be conveniently used to mark which lines are to be sorted by using the presence of a plus sign and a "some" sort. The plus sign can be removed prior to printing by use of the Arrange, Clean, or Replace commands depending on the circumstances.

SORTING ON MULTIPLE KEYS

Sorting a list on the basis of more than one key zone is accomplished by sorting first on the least important key, then the next most important, etc., until finally the most important key is used.

For example, to sort the list below by "Name" within "Sex," the command shown would be used.

	1	5	10	15	20	25
					

BEFORE	Name	Age	Sex
	John	25	M
	Mary	33	F
	Margret	15	F
	Tom	42	M
	Barbara	25	F

WHAT NEXT? SORT P 1 10 SORT P 22 22

	1	5	10	15	20	25
					

AFTER	Name	Age	Sex
	Barbara	25	F
	Margret	15	F
	Mary	33	F
	John	25	M
	Tom	42	M

THE TALLY COMMAND

The Tally command counts instances of like elements on a list in the working area or on disk. It can also calculate the totals of values within categories. The general form of the Tally command is:

```
WHAT NEXT? TALLY { IT or disk-file-name } { ALL }
           { key-left-position key-right-position, etc. }
           { SUM left-position right-position, etc. } END
```

Tally inserts a report of its results into the working area. The first line of the report simply shows the exact Tally command used to generate the tally. This is useful as a reminder. The next lines in the report give the frequencies of list elements falling within key zones.

Tally is useful for analyzing questionnaires or for running quickly through list data in order to determine frequency distributions or group totals. For example, Tally can be used on a list of checks to find out the number of checks per category and the total expenditure within categories.

THE LIST

The Tally command works on a list as its source of data. This list normally contains a number of lines which hold a variety of data items for a particular element or transaction. An example list is shown below.

	Sex	Status	Salary	Age	Job Type
	-----	-----	-----	-----	-----
	8-8	14-14	19--24	28-30	37-39
+	M	S	5500	23	PRO
+	F	M	1200	28	GOV
+	M	S	1600	45	SKL
+	F	S	1920	55	SKL
+	M	M	2150	59	SEM
+	M	M	1370	27	UNS
+	F	D	980	21	SKL

This list shows for a number of persons interviewed, their sex (male or female), their marital status (single, married, or

divorced), their monthly salary, their age, and their job type (professional, government, skilled, self-employed, or unskilled). Codes have been created to represent the various possibilities. The list was written out, of course, "in the editor."

Because Tally normally considers only lines which have a plus sign in the first position, plus signs have been placed in position 1 of each data line. This means, incidentally, that lists or tables with explanatory headings do not have to be modified before they are tallied--as long as the headings do not have a plus sign in position 1, Tally will ignore them.

Under the heading in the example, a line has been included to show in which physical positions the various data elements fall. This is a good practice, since Tally requires that zones of interest in the list be specified by the positions they occupy. Including such information in the list means it does not have to be figured out anew at each occasion Tally is used.

Tally is sometimes used on a table prepared for the Compute command. Usually, it is not hard to arrange a table so that the conventions of both Compute and Tally are obeyed. In fact, it would have been wise in the previous list to put in an example line and mathematical symbols to count all the lines and calculate the total of incomes. A use of Compute could then be made to derive control totals which could be compared to any results achieved with Tally.

IDENTIFYING THE LIST

The first parameter following the command-name identifies the list's name ("it", if the list is the working file). Tally will process lists in files on disk without first making such files the working file.

Thus, to tally the working area, the command would begin:

TALLY IT

To tally a file named "SURVEY" on disk, the command would begin:

TALLY SURVEY

THE "ALL" OPTION

Tally's second parameter is optional. As already noted, Tally works normally on lines which contain a plus sign in the first position. If the word "all" is given as the second parameter, then all non-blank lines will be tallied even if they do not begin with a plus sign. This option is useful when a list has only data and no headings or extraneous information.

When the "all" option is used to tally the working area, the command would begin as shown below.

```
TALLY IT ALL
```

THE KEY ZONES

The next set of parameters in the Tally command describe the zones of interest in the lines to be tallied. "Key Zones" are identified by the column-position of the left side and the column-position of the right side of the zone.

Tally commands always end with the word "end", which tells T/Maker that all desired parameters have been specified.

In the simplest use of Tally, only one zone is specified, followed by the word "end" to show that the list of desired parameters is complete. When Tally is used in this manner, it simply counts the number of times particular combinations of characters were found in the key zone.

For example, the command

```
TALLY IT 8 8 END
```

on the example list would produce

```
TALLY IT 8 8 END
```

```
F =      3  
M =      4
```

As mentioned above, the first line of the report simply shows the statement used to generate it. The next two lines say that an "F" was found in the key zone (i.e., position 8 through 8) three times, and an "M" was found four times. Thus, the results tell that three females and four males were interviewed.

Changing the definition of the key zone will obviously change the tally carried out. To tally Job Type, for example, the command below would be used:

TALLY IT 37 39 END

In many instances, it is desirable to tally combinations of key zones. Consider the command below.

TALLY IT 8 8 37 39 END

Here, two key zones are defined--Sex and Job Type. The resulting report is shown below.

TALLY IT 8 8 37 39 END

F:SKL =	2
F:GOV =	1
M:SEM =	1
M:SKL =	1
M:PRO =	1
M:UNS =	1

When more than one key zone is specified, a colon is inserted between the characteristic-names found in key zones. This report, for example, reports that two females had skilled jobs, one female had a government job, and so on.

The lines in a Tally report do not appear in any fixed order. However, the Sort command can be used to order it. (Note: a "partial" sort will insure that the replica of the command line is not sorted in with the Tally report.)

Sex	Status	Salary	Age	Job Type
=====	=====	=====	=====	=====
8-8	14-14	19--24	28-30	37-39
+	M	S	5500	23
+	F	M	1200	28
+	M	S	1600	45
+	F	S	1920	55
+	M	M	2150	59
+	M	M	1370	27
+	F	D	980	21

THE SUM ZONES

Sometimes it is useful to know not only how many items of a particular characteristic occur but also the total value of a variable associated with the characteristic. This is accomplished by terminating the list of key zones with the word "SUM" and following this word with the beginning and ending position of a variable upon which one desires sums calculated.

In the report below, marital status was tabulated while summing incomes for each of the groups (divorced, married, and single). The last line, for example, reports that "S" was found in position 14 three times and that the total of positions 19 through 24 for S's was 9,020.

TALLY IT 14 14 SUM 19 24 END

D =	1	980
M =	3	4720
S =	3	9020

The lines in a Tally report always show the values found in the zones, an equal sign, the number of occurrences of the value, and, finally, the totals of the zones summed in the order specified.

A report based on the same list, this time using two key zones and two sum zones, is shown below.

TALLY IT 8 8 37 39 SUM 28 30 19 24 END

F:SKL =	2	76	2900
F:GOV =	1	28	1200
M:SEM =	1	59	2150
M:SKL =	1	45	1600
M:PRO =	1	23	5500
M:UNS =	1	27	1370

When zones are summed, Tally always presents group totals in the minimum number of digits possible. So, for example, the following report might be generated:

Francs =	12	76.45
Dollars =	15	28.5
Lire =	7	1156859

Note that even though each sum is presented to a different accuracy, they are nevertheless vertically aligned.

LIMITS

The total number of zones specified (key or sum) may not exceed 10. It would be difficult to spell out precisely the size of tallying jobs that T/Maker can do. To be sure, there is a limit on job size. That limit is made up of a combination of factors--the total length of key zones, the number of zones to be summed, and the total number of possible combinations of the key zones--and so cannot easily be defined. As an example, Tally can handle a problem involving a total length of key zones of 20 positions, 3 zones summed, and 50 different possibilities. If the problem is too complex for Tally to solve, an error message will be generated. The easiest way to get around the problem is either to make separate passes for summing different values or to reduce the length of key zones.

USING TALLY WITH LOAD#

The report generated by Tally obeys the conventions about data files set down by the Load command. Thus, the Tally report's results can be saved and loaded into tables or other text files without having to perform an editing session. Consider the mask shown below.

```
=====
          Average Incomes By Various Categories
=====
      Number      Total Income      Average Income
=====
ex           999999       9,999,999      999,999.99
zv
ac1
+
+    Males      (>M )      (      )
+    Females     (F  )      (      )
+
=    Total
=====
+
+    Professional   (PRO )      (      )
+    Self-employed   (SEM )      (      )
+    Government Worker (GOV )      (      )
+    Office Worker    (OFF )      (      )
+    Skilled Worker    (SKL )      (      )
+    Unskilled Worker  (UNS )      (      )
+
=    Total
=====
```

* A knowledge of the Load command is required to understand the material presented here.

This mask was developed with names corresponding to the codes used in the list of data. In fact, an extra category of employment type (i.e. "OFF") has been anticipated. If the above file had the name "MASK" and the example list had the name "SURVEY", then the following command (if put on one line and executed with the Do command) would produce the result below.

```
DELETE TEMP CREATE TEMP
TALLY SURVEY 8 8 SUM 19 24 END
TALLY SURVEY 37 39 SUM 19 24 END
SAVE G MASK LOAD TEMP LOAD BLANK
COMPUTE CLEAN PRINT NONSTOP IT
```

The sequence creates a dummy file called "TEMP" into which the Tally command can put its reports. Two separate reports are generated for Income by Sex and Income by Job Type. This file of data is then saved. The mask is next established as the working file, and the data is loaded. A use of Load with the "Blank" option is done to replace any unresolved data names with blanks. Finally, Compute, Clean, and Print are performed.

If control totals of the list, itself, had been developed by using Compute, these could be compared to the table produced to insure that no data was lost through miscodings.

Average Incomes By Various Categories			
	Number	Total Income	Average Income
Males	4	10,620	2,655.00
Females	3	4,100	1,366.67
Total	7	14,720	2,102.86
Professional	1	5,500	5,500.00
Self-employed	1	2,150	2,150.00
Government Worker	1	1,200	1,200.00
Office Worker			
Skilled Worker	3	4,500	1,500.00
Unskilled Worker	1	1,370	1,370.00
Total	7	14,720	2,102.86

Expert's Tip: Since Tally generates a data file usable with the Load command, it is sometimes convenient to make use of Tally instead of Unload to extract data. Examine the table below and the report generated.

TABLE:

1	5	10	15	20	25	30
.....

REPORT:

ex	9,999.99	TALLY TABLE	14	21	SUM	24	31	END
+ Mortgage	500.00	Expenses =	1					1325
+ Car Payment	275.00	Income =	1					1780
+ Other	550.00							
+ = + Total Expenses	1,325.00							
+ Salary	1,480.00							
+ Consulting	300.00							
+ = - Total Income	1,780.00							
= Net Income	455.00							

TRANSFER UTILITIES

The commands described in this section permit the ad hoc movement of random elements of data from one file to another. This can be done manually, of course, by finding the item of interest in one file, remembering it, and editing it into the other file.

T/Maker, however, can automate the transfer of data. It provides three commands for this purpose--Unload, Load, and Transfer. The Unload command transforms text in the working file into a special form where each element of data has an identifying name. The Load command retrieves data in this special form from disk and places it into the working file. The Transfer command is used primarily to move a few numbers from one table to another without having to go through any intermediate stage.

Since the Load command is often used without a corresponding use of the Unload command, it is described first.

TABLE OF CONTENTS

THE LOAD COMMAND	TU-1
THE UNLOAD COMMAND	TU-8
THE TRANSFER COMMAND.....	TU-10

THE LOAD COMMAND

The Load command is used to transfer (or load) data from a file on disk into the working area. The command-name is followed by the name of the disk file to be used or the option "blank" (to load blanks).

WHAT NEXT? LOAD (disk-file-name or BLANK)

Load may be used in conjunction with the Unload command to transfer data between files. It can also be used on its own as a means of separating data from tables or forms that make use of the data. For example, it permits many tables to be prepared from a single file of data. Thus, when a value changes, only the file of data need be edited.

Load can also be used with the result of a Tally command to prepare summary tables from a list of observations or transactions.

The use of Load involves two files. The first is a "data file" which contains data in a special form and the second is a "mask file" which describes where and how data should be placed. Both of these files, are constructed with the principle that every element of data has a unique name.

When Load is used, the working area should be a mask, and the disk file specified as a parameter should be a data file. Load takes the data from the data file and places it in the mask.

DATA FILES

A data file makes an association between names and values. To do this, the name of a value is put on the left-hand side of an equal sign with its value on the right-hand side. If the value contains a blank, it must be surrounded by single or double quotes. If a line in a data file has no equal sign, it is considered to be just a comment. Below is an example data file.

```
My Personal Data File
-----
age = 45
name.of.wife = 'Agnes Smith'
```

The first lines of this data file are just comments, since they contain no equal sign. The other lines make an association between a name and a value. The data named "age" has the value "45", and the data named "name.of.wife" has the value "Agnes Smith".

Normally, a period or a colon is used as a separator in names that are made of a few parts. Spaces may be put in a name, however, they are ignored (e.g., "A B C" is the same name as "ABC"). A name may make use of any characters except those below which are reserved for special purposes.

{ } # < > ! = -

There is no particular relationship between a name and a value except as is useful for one's own purposes. Names and values may be placed anywhere on a line so long as the name is to the left of the equal sign and the value to the right of it. The following is a perfectly valid data file.

```
name.of.wife      =          42
                    age      =      'Agnes Smith'
```

Clearly, however, it makes sense to pick meaningful names and to organize a data file so that it is easy to read and revise. It also makes sense to pick short names, if possible, for reasons to be discussed later.

A line of data may contain more than one value even though it contains only one name. In such cases, the values must be separated from each other by one or more spaces.

When this type of assignment is used, an assumption is made about the names of the other values on the line. The assumed name will always be "one higher than the last name used" (see next subsection).

For example, the two data files below are identical from the point of view of the associations they make between names and values.

sales.78 = 32,000	sales.78 = 32,000 43,000 51,000
sales.79 = 43,000	
sales.80 = 51,000	

"ONE HIGHER THAN THE LAST NAME USED"

The notion of a name being "one higher than the last name used" needs some clarification. If a name ends with a number, then one higher means adding one to the number. If a name does not end with a number, then one higher means appending the number "1" to the name. The table below shows some names in conjunction with names that are one higher.

<u>Last Name Used</u>	<u>One Higher</u>
12MX	12MX1
Y99	Y100
Y099	Y100
FFF0	FFF1
FFF	FFF1
X1.9	X1.10
C10,999	C10,1000

Note that Y99 and Y099 are both followed by the name Y100 even though they are not the same name. Notice also that a decimal point or a comma is not treated as a part of the number in a name (e.g., C10,999 is not followed by C11,000, but C10,1000).

The most common mistake in setting up a data file is to inadvertently have the same name represent two different data items as is done below. This happens because care is not taken in picking names or because values with blanks are not enclosed in quotes.

In the data file below, "cus1" is assigned the value "Bob" by the first line. This line also assigns the value "Smith" to the name "cus2". Notice that this conflicts with the assignment made by the second line. In cases like this, the first value assigned to a name is the one that will be used by Load.

```
cus1 = Bob Smith
cus2 = Doug Jones
```

The error above could be avoided by either of the approaches below.

```
cus1 = 'Bob Smith'
cus2 = 'Doug Jones'
```

```
cus1.1 = Bob Smith
cus2.1 = Doug Jones
```

MASKS

When Load is used, the working area is expected to be a "mask." A mask is a sort of map that describes where data is to be placed and how it is to be placed.

To show where data should be placed in a mask, braces are used to define the appropriate zone on a line. For example, the braces below define the zone (on the line they are on) that extends from the open brace through the closed brace. Note that it is impossible to define a zone which is only one character wide, since it takes two characters to define the zone.

{ }

To show how data should be placed, a special character is put inside the braces.

{< >}

The special characters which may be used and their effects are noted in the table below.

<u>Character</u>	<u>Interpretation</u>
-	The data should be centered in the zone.
>	The data should be lined up with the rightmost limit of the zone (right justified).
<	The data should be lined up with the leftmost limit of the zone (left justified).
!	The data should be lined up with the leftmost limit and unused space should be eliminated after the data (squeezed).

To show what data should be put in the zone, a name is put inside the braces.

{<name.of.wife >}

The same name as in the corresponding data file should be used. If no name is put inside the braces, the name will be assumed to be one more than the last name used as with data files. In this case though, the last name used is the name in the previous set of braces.

Note: Making a big mask in which a few hundred variables are called for can appear to be a terrible editing task. However, a skilled use of editing features like the keystroke macro, copying lines and columns, and replacing strings, often means such a mask can be welded together in a few minutes.

LONG NAMES IN MASKS

In some cases, the name of a variable may be longer than the space to be used for its value. While it is best to avoid this situation by picking short names, there is a means of putting a long name in a short zone. Consider the mask below.

{ } { }

Suppose the names "sales.75" and "sales.76" are to be placed in the two locations shown above. Clearly, there is not enough space as is. The correct procedure under these circumstances is the following.

- First put the braces appropriately to define the zones in which all the values for the line are to appear.

{ } { }

- Next, put the cursor just inside the open brace.

{ } { }

- Begin typing the special symbol and the name of the variable until all the space is used (i.e., the cursor is over the close brace).

(>s#) { }

- Set Insert Mode in the editor, type a "#" and then the rest of the name.

(>s#ales.75) { }

- Clear Insert Mode.

The number sign is used to show the spot where the close brace would have been placed if the space had been sufficient. In the current example, sales.76 need not be entered into the second set of braces, since it is the next name after sales.75. If the name were something else, however, the same technique could again be used.

When data is loaded into such a zone, the space from just after the number sign through the close brace is eliminated. Other zones to the right are thus pulled to the left.

SOME NOTES ON THE USE OF LOAD

The Load command can only be used when the working file is a mask. The command expects a single parameter which should be the name of the data file or the word "blank".

As Load reads the data, it attempts to complete the mask. If a name is associated with two different values in the data file, the first value found in the data file is always used.

If a set of braces does not contain a special symbol to show which type of justification is required (ie. < > ! or -), the justification used for the previous set of braces is used again. If there is no previous set, the default is left-justification.

If the data value is too big to fit in the space provided, it is not inserted and the braces are left as they are. If the data file does not contain any data for a particular reference, the braces are left. In the case that the braces are not replaced by the corresponding data, the appropriate name is put inside the braces. Consider the mask below.

```
{cost.75} ( ) {sales.75} ( )
```

Suppose the data file only contained data for cost.75 and sales.75, but not cost.76 or sales.76. After loading the data, the remaining mask might look like the following.

```
134.23 {cost.76} 241.67 {sales.76}
```

If the data for 1976 were in another file, it could then be loaded without having to worry that some empty braces no longer corresponded to the appropriate name.

When the name "blank" is given instead of a file name, any unfilled braces are replaced by blanks.

A count of the number of data values which did not fit in the space provided for them and a count of the number of braces for which no data could be found are displayed on the screen after the Load command has been completed.

AN EXAMPLE OF LOAD

A Data File (on disk)	A Mask (the working area)	
Address Information	Name: (<name)
company = 'Doe & Sons, Inc.'	Address: (<street)
name = 'John Doe'	(<city)
street = '1431 West 45th Street'	(!state) (!zip
city = 'New York')
state = NY	(-company)
zip = 10028		1980 1981
Sales Figures	Sales: (>sales.80) ()
sales.80 = 20,000 21,000		

**The Result of Load
(the new working area)**

Name: John Doe

Address: 1431 West 45th Street
New York
NY 10028

Doe & Sons, Inc.

	1980	1981
Sales:	20,000	21,000

THE UNLOAD COMMAND

The Unload command moves (or "unloads") data from the working area to create a new working area which is a data file.* Thus, the result of Unload can be saved on disk and used with the Load command to place the data in some other text file.

WHAT NEXT? UNLOAD disk-file-name

The file on disk must be a mask.* Unload converts the working file from a text file to a data file on the basis of the mask. The original working area is destroyed in the process, so it is good practice to rename the file before unloading it. Unload provides a count of the number of lines in mask file that did not match lines in the working area and were therefore not unloaded.

THE MASK FILE

When using Unload, a mask shows where data are located in the text file and the names of data items to be unloaded. Braces on a line in the mask mean that the corresponding line in the text file should be unloaded. If a line in the mask has no braces, it is ignored.

The lines in the mask must be in the same order as those in the text file. And, the mask line must match the text line, character for character, for all positions to the left of the first brace on the mask line. This brace can be either an open brace which would signal the first value to be unloaded or a close brace which simply demarks the end of the matching zone.

To prepare a mask for the purpose of using Unload, the easiest way is to start with the text file to be unloaded giving it a new name. Make sure that each line in the text file to be unloaded begins with a string of characters suitable for matching purposes.

Next, delete with the editor all lines that are not to be unloaded. Then, put braces around the zones to be unloaded and put the names of the data items inside the braces. If necessary, end the matching zone before the first open brace by placing a closed brace on the line.

* See the Load command first for more information about masks and data files.

AN EXAMPLE OF UNLOAD

A Mask File (on disk)	A Text File (the working area)
This mask unloads name and 1981 Sales	Name: John Doe
Name: (name)	Address: 1431 West 45th Street New York NY 10028
Sales:) (sales.81)	Doe & Sons, Inc.
	1980 1981
	Sales: 20,000 21,000
	A Data File (the new working area)
	name = 'John Doe' sales.81 = '21,000'

Tip: The only purpose of using Unload is to create a data file from a text file so that the data can be transferred into other files. Often, however, a "data file" can be created in other ways without the need for a mask. Consider the line below which might have appeared in a table.

= / Grand Total 12,765 14,199

The Clean command could make this line into the one below.

Grand Total 12,765 14,199

The Merge command could then be used to superimpose a file which consisted of lines with just equal signs in the correct position creating:

Grand Total = 12,765 14,199

Note that this is a valid data line and equivalent to "GrandTotal = 12,765" and "GrandTotal1 = 14,199" (spaces in a name do not count).

There is no rule that data files must be constructed using Unload. A dash of creative fiddling sometimes does the job better. (See also the tip at the end of the description of the Tally command in List Processing.)

THE TRANSFER COMMAND

The Transfer command transfers data from the disk file into the Working Area. It is useful for integrating numbers from one table into another. It reduces the normal "UNLOAD SAVE GET LOAD" sequence to a single step that requires no additional files.

WHAT NEXT? transfer { disk-file-name or blank }

MASK LINES

When using the Transfer command, the disk file and the Working File both contain mask lines like those used for Load/Unload. However, these lines are placed immediately ahead of the line in the file they are associated with and not in a separate file as with Load/Unload.

```
= Total Expenses           {expenses }  
                           .....
```

In the above example, "expenses" would be known as the value occupying the zone shown as periods. As with masks used with Load or Unload, a number sign can be put within the braces to deal with names that are longer than the data zone they define (see below).

```
= Total Expenses           {expe#nses}  
                           .....
```

Note in the above that these two lines could be in the file on disk and would therefore define a data value which might be transferred into the Working Area. Also, however, they could be in the Working Area and would therefore define where a value should be placed.

SPECIAL SYMBOLS

When a value is to be placed in the working area, one of three symbols is used within the braces to specify how the values should be placed. These symbols are used in a manner similar to the way they are used with the Load command:

- < to left justify data in the working area,
- > to right justify data in the working area, and
- to center data in the working area.

Unlike the Load command, however, every set of braces must have one of these three symbols if the set of braces is intended to define a zone into which data should be transferred. Putting in none of the three symbols means the zone is "protected" and cannot be changed by data being transferred.

The symbol "!" is used to define a zone which is suitable as a source of data on disk. Without such a symbol, it is assumed the zone defined is intended only to be used in the working area as a destination for data. Thus, in order for data to be transferred from disk to the working area:

- 1) There must be corresponding names defined in both the disk file and the working area.
- 2) The zone in the working area must contain the symbol ">", "<", or "-".
- 3) The zone in the disk file must contain the symbol "!".

As with Load and Unload, a set of braces which contains no name is assumed to have a name "one higher than the last name used." However, no assumption is made about the various symbols within it.

Unlike Load, the braces are not replaced by the data because the data goes into the next line (a "DROP (" could be used to remove them or they could be presented as print comment lines to avoid printing them).

THE BLANK OPTION

When "BLANK" is specified in place of a file name, all data zones in the working area which are potential destinations for data are set to spaces (i.e., all but those which are protected).

OVERSIZED VALUES

If a value is too big to fit in the space allotted, the space is filled with asterisks. A count of values which overflowed the space for them is printed as a warning message.

AN EXAMPLE

Suppose that we keep our business expenses as a table like the one below.

Expenses		
	Item	Amount
ex		999,999.99
+	Copies	14.95
+	Pencils	11.23
..		{!expenses}
=	Total	26.18

Whenever we add items to the list, we can recompute the table to get the correct total. Notice that a mask line is put ahead of the total line with only the symbol "!!". This ensures that this total can be transferred into other files. Since we did not put the symbols "<", ">", or "-", no data can be transferred into the zone. That makes sense, since this table is the correct source of this data. Notice also that the mask line has been made a Print Design Comment Line so that it will not show up should we print the table.

Suppose we also keep a second table showing the performance of our business like the one below.

```

-----  

Income and Performance  

-----  

Description Amount  

-----  

ex 999,999.99  

+ Lawn Mowed 10.00  

+ Car Washed 8.50  

-----  

= + Total Income  

-----  

.. (>expenses)  

- Expenses  

-----  

.. (!net )  

= Net Income  

-----
```

We can add new sources of income to this table whenever we want; however, before computing it, it would make sense to transfer the total for expenses from our file of expenses so that we will have the most recent information. Notice in the mask line above "expenses" that the symbol ">" has been used which says that data should be right justified in the zone. Notice also that we have defined Net Income as a value in this table which might be transferred into some other table.

While it is not used here, a zone can be defined as suitable for transferring into and transferring out of as below:

```
{>!ratio }
```


DATA ENCRYPTION

THE MIX COMMAND

T/Maker's MIX command allows you to encrypt or code the working file. This command works on the whole working file, not just the active area.

The same command word, MIX, also serves to decode a working file previously encoded by MIX.

You use the MIX command by typing MIX and a carriage return after T/Maker's WHAT NEXT?-prompt.

WHAT NEXT? MIX

T/Maker will then prompt you for a password or clue, like this:

ENCODE WITH WHAT PASSWORD/CLUE?

T/Maker is asking you to specify either a password alone or a password and a clue. What are passwords and clues? How are they used?

Passwords

Think of the MIX command as locking up the working file's contents in a tightly sealed box of codes.

A password is a word that acts as a key to that box. You specify a password when you encrypt the working file in the first place. Later, you will need this same password to decode the working file and return its contents to plaintext.

Suppose that you use the password, SHAZAM. Simple passwords must not contain blank spaces. You may, however, include blank spaces in a password string if you surround the password string (including the clue, if any) with single or double quotation marks (e.g., "hocus pocus/magic"). (Note: case is important in passwords, thus "SHAZAM" and "Shazaam" are not the same password.)

Once SHAZAM has been typed-in and the carriage return key has been pressed, T/Maker will encode the file. An encrypted file will appear on the screen.

If you now save the encrypted working file, it will be saved under the working file's current filename.

Once the file has been encrypted, NEVER go into the T/Maker editor or attempt to edit an encrypted file. T/Maker's method of coding your file is sequential--which is to say, the code for a single letter is dependent upon the code for the letter preceding it. Hence, altering an encrypted file will make that file permanently un-decodeable for T/Maker. NEVER use T/Maker's INSERT, MERGE, COMPUTE, CLEAN, or any other command that does work upon a T/Maker file upon an already encrypted file.

You will need the SHAZAAM password to decode the encrypted file when you wish to.

Clue Words

Clue words serve the function of memory joggers. The clue word you provide will appear at the top of the encrypted file. The clue word appears there in order to jog your memory for the correct password. You nominate a clue word by following your password with a slash and then typing the clue word itself. There are no spaces in this series--so, for example, the password/clue word sequence SHAZAAM/CAPTAIN would be typed in as follows:

ENCODE WITH WHAT PASSWORD/CLUE? SHAZAAM/CAPTAIN

This clue word, "CAPTAIN," in this case might have been used in order to remind you that it was Captain America who always said "Shazaam."

You may introduce a clue word each time you encrypt a file. However, you are not required to nominate a clue word--T/Maker will be satisfied if you merely provide a password for the MIX command.

Other Considerations

1) Encrypting a file protects you from having other people read your file. It does not protect you from having other people destroy the file.

2) T/Maker normally maintains one level of backup with a ".BAK" file. Thus GET-ting an unencrypted file, MIX-ing it, and saving it will result in a perfectly readable ".BAK" file. You should save your file twice in a row to insure that the backup file is also in the encrypted form or you could delete the appropriate ".BAK" file.

3) If you make a mistake and type your password incorrectly, but only realize it after you have touched RETURN, you should make the same mistake to decode the file.

4) Your password is not entered into your file in any way. If you forget your password, you may consider the file permanently lost.

DESIGNING SYSTEMS WITH T/MAKER

Many tasks involve repetitive operations. In such cases it may be desirable to set up a structured approach. It may also be necessary at times to have people who really know very little about T/Maker using T/Maker. In both these instances, the commands and concepts covered in these sections will be helpful.

Typically, a "System" amounts to one or more database files and a number of special report or support files. More often than not, the HELP command with special screens and predefined commands is used to develop any easy menu-type interface, and the Update command is used to enter the raw data. From there a series of files and automated command sequences are used to move data about and generate any required reports.

TABLE OF CONTENTS

THE DO COMMAND	DS-1
THE HELP COMMAND	DS-4
INTERFACING WITH OTHER PROGRAMS	DS-11

THE DO COMMAND

T/Maker's DO Command lets you convert lines of text lodged inside a file into a string of command responses to the WHAT NEXT?-prompt.

Because the DO command permits you to convert ordinary text into T/Maker commands, DO is T/Maker's passport to programming. The DO command, in other words, lets you store series of commands for later use.

You use this command by typing "DO" in response to the WHAT NEXT?-prompt and pressing the carriage return key. T/Maker hears this command as follows:

"T/Maker, first, please go to the top line of the screen (or top line of the frame, if you are in Frame Mode) and grab whatever text that appears there. Second, regard that text as a series of commands--that is to say, place that text after the WHAT NEXT?-prompt, and process the commands as you would any others."

A line of text inscribed at the top of a file and intended for use as a string of commands is called a "DO COMMAND LINE" or "DO LINE" for short.

A simple DO LINE might look like this on your screen:

SCREEN 1:

```
-----  
| ALIGN SAVE PRINT NONSTOP IT  
|  
| <<<                                ->>  
| Now is the time for all good men to come to the aid of  
| their party. Now is the time for all good men  
| to come to the aid of their party.  
|  
| 1/1 WHAT NEXT? DO  
-----
```

SCREEN 1 shows the situation just before you press the carriage return key. Once the carriage return key is pressed, the screen will--for an instant--look like that shown in SCREEN 2.

SCREEN 2:

```
|  
| <<<                                ->>  
| Now is the time for all good men to come to the aid of  
| their party. Now is the time for all good men  
| to come to the aid of their party.  
|  
| 1/1 WHAT NEXT? ALIGN SAVE PRINT NONSTOP IT
```

T/Maker will now process these commands just as if they had been entered in the conventional way. No further carriage return is required. Thus, in a moment the screen will look just like SCREEN 3's, below:

SCREEN 3:

```
|  
| <<<                                ->>  
| Now is the time for all good men to come to the aid  
| of their party. Now is the time for all good men  
| to come to the aid of their party.  
|  
| 1/1 WHAT NEXT?
```

Multiple DO LINES

DO LINES may be as long as any line of text--that is, 300 or 400 characters in length depending on your hardware. You may have need for an even longer DO LINE or for a command system that breaks a task into two or more subtasks.

T/Maker can easily accomodate this need. All you need do is inscribe a series of DO LINES--one beneath another, beginning on the top line of the screen. You must end all such DO LINES except the final one with the word DO. This "DO," then, will send T/Maker to the current top line on the screen, grab that line of text, and lodge it as a new series of T/Maker commands. For example, the following DO LINES would perform precisely in the aforementioned way:

```
COMPUTE SAVE DO  
CLEAN PRINT NONSTOP IT
```

DO LINES Lodged on a Series of Files

You may also automate T/Maker commands over a series of files. For example, you might lodge one DO LINE at the top of File A and another DO LINE at the top of File B. The DO LINE at the top of FILE A might end with the command words, "GET B DO."

This command will, of course, serve to make File B the working file and grab its DO LINE, converting that line into a series of commands.

Establishing DO FILES. You may also construct files that consist entirely of DO LINES--these are called DO FILES.

Suppose that you created a file called COMMAND and lodged the following three DO LINES in it:

```
GET TABLE.1 COMPUTE CLEAN PRINT NONSTOP IT GET COMMAND 2 DO  
GET TABLE.2 COMPUTE CLEAN PRINT NONSTOP IT GET COMMAND 3 DO  
GET TABLE.3 COMPUTE CLEAN PRINT NONSTOP IT
```

You saved this file on disk. Once thus saved, you can ask T/Maker to carry out the tasks described in these DO LINES by giving the instruction "GET COMMAND DO" in response to the WHAT NEXT?-prompt.

Notice how T/Maker would go about its assignment: First, it would make the file named COMMAND the working file. Next, it would grab the top DO LINE and carry out the tasks this line requests. When T/Maker reaches the last part of this line--i.e., "GET COMMAND 2 DO"--T/Maker will (1) once again make the COMMAND file the working file, (2) move the top of the screen to the file's second line (the "2" command does this), and (3) grab the line thus placed at the top of the screen and make it into a series of T/Maker commands.

Once the second DO LINE's work is complete, T/Maker would follow the same sort of procedure in carrying out the third DO LINE.

This method, then, provides a convenient device for getting T/Maker to work its way through a series of commands all by itself.

THE HELP COMMAND

MAKING UP YOUR OWN T/MAKER COMMANDS

T/Maker lets you invent your own commands. You can invent commands that present INFORMATION on the screen and/or cause ACTIONS to take place. The information which is displayed and the action which takes place can be made a function of what the user types. In short, you can make MENU-type systems.

CONSTRUCTING NEW COMMANDS:

You need to construct two files to make a new command.

The first file is called the NAME FILE. The second file is called the CONTENT FILE.

Both the NAME file and the CONTENT file must be lodged on the disk drive or sub-directory (for hard disk machines) that holds your other T/Maker command files (for example, EDITOR.TMK, ALIGN.TMK, etc.).

(1) THE NAME FILE:

Your NAME file is an exact copy of T/Maker's HELP.TMK file with a new name. The new name consists of your new command's name followed by ".TMK" ("/.TMK" for TRSDOS).

You must use your operating system functions to copy T/Maker's HELP.TMK file, renaming it in the process.

If your new command were named "ASSIST" then you would copy "A:HELP.TMK," renaming it "A:ASSIST.TMK".

Obviously you will want to pick a name for your command that isn't already in use for another T/Maker command. Also, don't start your command word with an "E." Command words can be up to eight characters long.

(2) THE CONTENT FILE:

Your CONTENT file consists of the actual content of your command.

The CONTENT file must be named in a particular way. It must be named with the command name followed by ".1" ("/.A" for TRSDOS).

If, for example, your new command were named "ASSIST" then your CONTENT FILE for it would be named "ASSIST.1". ("ASSIST/A" for TRSDOS).

Your CONTENT file is comprised of a number of things.

- a) Your own PROMPT. This prompt will appear on your screen when your command is invoked.
- b) Your INDEX LINE(S). This line or lines tells T/Maker the legitimate TOPICS and where they are.
- c) Your INITIAL SCREEN. This is the first screen to show. It usually shows a number of options to pick from.
- d) Your OTHER SCRRENS. These panels hold the actual informational material associated with each topic.
- e) Your ACTION LINES. These lines describe the actions to be taken in response to particular words typed by the user.

The editor is used to create and place all these elements in your CONTENT FILE.

THE STRUCTURE OF YOUR CONTENT FILE

The Prompt

The file's TOP LINE holds your PROMPT. Your prompt might, for example, read:

WHAT DO YOU WANT TO LEARN ABOUT (OR END)?

Always include an "(OR END)" in your prompt. It serves to remind you or another user that you must type END to exit your command.

The Index Line

The file's SECOND LINE holds your INDEX LINE. This line contains three things in the following order:

- (a) An "=" sign in column position one.
- (b) The file's name.
- (c) Finally, a list of the options your informational command will offer.

All of these elements are separated by single blank spaces.

So far, then, the exemplary ASSIST.1 file might look like this:

```
|-----|  
| WHAT DO YOU WANT TO LEARN ABOUT (OR END)? |  
| = ASSIST.1 ALL WORDS MATH LISTS DATA |  
|-----|
```

It is possible to have more than one CONTENT FILE associated with your command. In such instances, you would have one INDEX LINE for each CONTENT FILE. Subsequent INDEX LINES, like the first, would show the name of the addition file and the TOPICS in it. Below is an example with three CONTENT FILES.

```
|-----|  
| WHAT DO YOU WANT TO LEARN ABOUT (OR END)? |  
| = ASSIST.1 ALL WORDS MATH LISTS DATA |  
| = ASSIST.2 SPELLING GRAPHS |  
| = ASSIST.3 OTHER |  
|-----|
```

You may give CONTENT FILES other than the first any name you like. Keeping a numbering convention like that used above would seem, however, a simple and adequate approach.

Screens and Header Lines

The area beneath your file's PROMPT and INDEX lines provides the area for your INITIAL SCREEN.

All screens, however, begin with a HEADER LINE, which, in turn, is comprised of two elements: a LEVEL NUMBER and a TOPIC WORD. The HEADER LINE signals T/Maker that the text appearing below it is the screen for a given TOPIC.

A HEADER LINE consists of a plus sign followed by a LEVEL NUMBER ("1" in this case) followed by a TOPIC WORD (in this case, "MENU"). Single blank spaces separate these three elements.

The exemplary file with its HEADER LINE in place might look like this:

```
|-----|  
| WHAT DO YOU WANT TO LEARN ABOUT (OR END)? |  
| = ASSIST.1 MENU WORDS MATH LISTS DATA |  
| + 1 MENU |  
|-----|
```

Notice that the TOPIC WORD on the header line (i.e., "MENU") should appear on the appropriate INDEX LINE. This insures that a user may type "MENU" to the prompt to force the first screen to

reappear. It is often a good idea to mention how to get back to the first screen in the prompt.

```
|-----|  
| WHAT DO YOU WANT TO LEARN ABOUT (OR MENU OR END)? |  
| = ASSIST.1 MENU WORDS MATH LISTS DATA |  
| + 1 MENU |
```

You construct the SCREEN below the line holding the HEADER LINE. You can construct this panel as you like, using all of the powers of the editor. The only requirement you must follow is that the topic words ("WORDS," "MATH," "LISTS," and "DATA") in your HEADER LINE must be typed exactly (except for case--upper or lower) as they were on the INDEX LINE and in the order that the subjects appear in the file.

```
|-----|  
| WHAT DO YOU WANT TO LEARN ABOUT (OR MENU OR END)? |  
| = ASSIST.1 MENU WORDS MATH LISTS DATA |  
| + 1 MENU |  
| ====== |  
| FOR INFORMATION |  
| ABOUT: |  
| ====== |  
| Word Processing and Printing | WORDS |  
| Electronic Spreadsheet | MATH |  
| List Processing Functions | LISTS |  
| Database Functions | DATA |  
| ====== |
```

In addition to your INITIAL SCREEN you will want, of course, to enter the HEADER LINES and TEXT for the other screens.

Each of these descriptions is laid out in the file below the menu screen. Each, as well, is begun with "+ 1" in columns one through three followed by a blank space and the menu command word.

Thus, for example, the following screens might be crafted:

```
|-----|  
| WHAT DO YOU WANT TO LEARN ABOUT (OR MENU OR END)? |  
| = ASSIST.1 MENU WORDS MATH LISTS DATA |  
| + 1 MENU |  
| ======  
| FOR INFORMATION | TYPE THIS WORD |  
| ABOUT: | AND PRESS RETURN: |  
|-----|  
| Word Processing and Printing | WORDS |  
| Electronic Spreadsheet | MATH |  
| List Processing Functions | LISTS |  
| Database Functions | DATA |  
|=====|  
| + 1 WORDS |  
|-----|  
| Word Processing involves three T/Maker commands:  
| EDIT, ALIGN, and PRINT.  
|-----|  
| Word Processing also draws upon "Design  
| Commands," which are entered into a file as text.  
|-----|
```

Action Lines

You can add ACTION LINES to any screen. These lines can cause actions to take place in response to words typed by the user. For example, let us suppose that we want a user to be able to type PRINT as a response to the screen for the WORDS. If he does, we would like T/Maker to go back to the top of his file, align the text and print it. This can be accomplished by adding the following ACTION line below the HEADER LINE.

- PRINT 1 ALIGN PRINT IT

An ACTION LINE consists of a minus sign in the first column followed by the correct user response ("PRINT" in this case), followed by the T/MAKER commands that should take place. You can have many action lines for a single screen. Of course, it makes sense to describe in the screen the hidden action which the user can invoke, since ACTION LINES like HEADER LINES do not appear on the screen. For example, we might make our WORDS screen look like the following.

```
+ 1 WORDS
- PRINT 1 ALIGN PRINT IT

Word Processing involves three T/Maker commands:
EDIT, ALIGN, and PRINT.

Word Processing also draws upon "Design
Commands," which are entered into a file as text.

Type PRINT to ALIGN and PRINT your entire file.
```

Notice the placement of the ACTION LINE just below the HEADER LINE and the last line added to the screen which suggests what the user might type to undertake the action.

The MORE Prompt

If you design a screen for your command which requires more lines than available on your console, your command will automatically put the following prompt on the screen when the screen is full.

MORE (Y/N) ?

You may respond with:

"Y" to see the next screen.
"N" to return to the normal prompt

You may control the exact line at which the more prompt will appear. This is done by adding a line with only a plus sign in the first column and placing it where you want the break to occur. The example below shows the proper technique.

```
+ 1 MENU
      First screen of text
+
      Next screen of text
```

Level Numbers

Up to now, we have always used a LEVEL NUMBER of one on all HEADER LINES. Other LEVEL NUMBERS may however be used to give some structure to a CONTENT FILE.

LEVEL NUMBERS can be used to control which screens are presented for a given subject. This is best shown by an example. Consider the CONTENT file below.

```
+ 1 WP          screen lines
+ 2 EDIT        screen lines
+ 3 KEYS        screen lines
+ 2 ALIGN       screen lines
+ 1 SS          screen lines
```

With the above levels, a request for help on "WP" would continue to show all following subjects until the subject "SS" which has a level less than or equal to the initial level. A request of EDIT would potentially show EDIT and KEYS. A request for KEYS would just show KEYS.

Thus, LEVEL NUMBERS provide a way to structure the screens which your command will display. This is convenient when one topic, in a sense, is "contained" in another. This structure technique, however, can only be used for subjects in the same CONTENT file.

GENERAL ADVICE

Don't make CONTENT files so long that it takes forever to get to the last subject directly.

Don't make each TOPIC a different CONTENT file.

The best approach will depend on the speed of your disk drives.

Remember that decoration and indenting everything to the center takes up space on disk and slows the presentation.

Since lines in a CONTENT file that begin with a plus, minus, or equal sign have a special significance, it is good practice to start all the text lines for your screens in column 2 or later.

INTERFACING WITH OTHER PROGRAMS

USING T/MAKER FILES WITH OTHER PROGRAMS

T/Maker files are relatively easy to interface with other programs. All T/Maker files are standard ASCII text with no embedded control characters other than a Carriage Return and Line Feed. Since the High-Bit character can be used to place characters in the file above the standard ASCII range, you should be careful to insure that these characters, if you put them in, do not conflict with the expectations of the other program.

One instance in which a T/Maker file may be non-standard is if the TABS option is in force while a file is being saved. In this case a special line is added to the head of a file. This line can be removed by GET-ting the file, specifying NOTABS, and saving the file.

USING FILES FROM OTHER PROGRAMS WITH T/MAKER

T/Maker accepts a standard ASCII file as input. MSDOS, and CP/M 86 versions of T/Maker will accept Tab characters in a file and expand them to every eight positions. The CP/M and TRSDOS versions will not accept Tab characters. Under CP/M, the PIP command can be used with the T Option to expand tabs before attempting to use the file with T/Maker. (Consult your CP/M manuals for details on the PIP command.)

If you want to use files from other programs with T/Maker, your first objective should be to generate a standard ASCII file. Since T/Maker may use a different approach to a problem than another software package, you should consider in which form you want to generate the data. For example, to interface with a cell-based spreadsheet, you may find it more useful to generate a disk image of the printed table which could be edited with T/Maker rather than generating a DIF file which has no particular significance in T/Maker.

T/Maker's Database is easy to interface. If your other program allows you to store data in fixed locations (e.g., "name" always located between column-positions 10 and 30), you need only add an appropriate record definition to the head of the file to use T/Maker's database commands. In fact, your record definition could be placed in a separate file with a "<CONTINUE>" command at its bottom to link the file created by the other program.

The use of the LOAD command with masks also represents an easy way to absorb data from other programs into a T/Maker file. This approach, however, requires that the other program generate a suitable data file for LOAD. When the other program does not, some simple tricks using the MERGE Command or perhaps a Keystroke Macro can often shape any file into a suitable form.

CUSTOMIZATION NOTES

The notes describe how to install T/Maker for unusual machines or how to customize it to suit individual tastes.

TABLE OF CONTENTS

GETTING T/MAKER TO WORK	CN-1
PROBLEMS AND ANSWERS	CN-6
CUSTOMIZING WITH T/MODIFY	CN-7
[1]* Standard Terminals.....	CN-8
[2] Special Characters Used By T/Maker.....	CN-8
[3] Changing Default Processing Options	CN-10
[4] Printer Output Control	CN-13
[5] The Number Of Rows And Columns On The Console....	CN-14
[6] Console Control.....	CN-14
[7] Cursor Addressing	CN-15
[8] Reverse Video Control	CN-16
[9] Changing Editing Keystrokes	CN-17
[10] Printer Output Technique	CN-20
[11] Console Input Technique	CN-21
[12] Console Output Technique	CN-22
[13] System Patches	CN-22
APPENDICES	
ASCII Codes and Characters (Decimal Equivalents).....	CN-24
Console Control Sequences	CN-25

* Numbers in brackets are the corresponding T/Modify option.

GETTING T/MAKER TO WORK

You have probably gotten to this section because you have received a GENERIC version of T/Maker which must be configured to work on your machine, or because you want to customize T/Maker to suit your preferences.

It is impossible to include a complete description of everything that might go wrong while trying to set up T/Maker on all computers. These notes may help you through a number of sticky situations. Some parts, however, are unavoidably technical. If you find yourself with a problem and no idea of how to solve it, call your dealer or distributor or write to T/Maker, 2115 Landings Drive, Mt. View, California 94043. If you phone, please try to have your computer handy with the appropriate files on disk. Also try to have your hardware manuals by your side.

Even if everything works correctly, you should consider reading these notes completely at least once. They provide information which can save both time and effort when using T/Maker.

MAKING BACKUP DISKS OF T/MAKER

The first thing to do is always to make copies of your T/Maker disks for backup purposes. Your operating system manuals will tell you how to do this.

In some cases, T/Maker is delivered in a disk format that your machine can read but cannot write. For these cases, you must use the facility of your operating system that allows you to copy individual files from one disk to another, rather than the complete disk duplication command.

DISKETTE ORGANIZATION

You may also want to prepare a disk which can both boot your computer and be used to run T/Maker. Your operating system manuals will tell you how to make an almost blank diskette that can be used to start-up your computer.

You can then use the facilities of your operating system to copy various T/Maker files to that diskette.

To start T/Maker you must have the the following files on your diskette.

OPERATING SYSTEM

CP/M80	CP/M86	MSDOS or PCDOS	TRSDOS
--------	--------	-------------------	--------

TMAKER.UTL	TMAKER.UTL	TMAKER.UTL	TMAKER/UTL
TMAKER.COM	TMAKER.CMD	TMAKER.EXE	TMAKER/CMD
TMAKER2.COM			EDITOR/TMK
EDITOR.TMK			

These files alone will permit you to create, save, and edit files as well as perform a number of utility operations. Major T/Maker commands are their own files and you need only have them available if you intend to use the corresponding command.

For example, if you wish to prepare a diskette to be used exclusively for Word Processing, you need not put the T/Maker file COMPUTE.TMK on it, since this file is used for the COMPUTE command.

You may switch diskettes while running T/Maker if you have the commands spread over more than one diskette. The files beginning with "TMAKER" are in fact not needed by T/Maker once it is already running. If you intend to do this while running under CP/M 80 or CP/M 86, be sure to pick option 3 of T/Modify and answer "YES" to the question "SHOULD A RESET BE DONE BEFORE EVERY SAVE."

For systems requiring more than one T/Maker diskette, we attempt to organize the files in the most useful way. You may find, however, that another organization is better suited to the type of tasks you wish to accomplish. It is also possible to invoke a command which is resident on a different drive by preceding the command name with a drive designator (e.g. B:COMPUTE -- not necessary for TRSDOS).

NAME CHANGES [1]

You may change the file-names of T/Maker Commands whose file-names end in ".TMK". This is done with the renaming command of your operating system. One restriction applies, however: you may not change a file-name to a file-name that begins with an "E".

[1] This section does not concern the initialization of T/Maker. for those purposes. It is written in response to a number of questions by T/Maker users.

Here is a list of shorter file-names you might wish to employ for some common commands.

Current	Short Name
-----	-----
ALIGN.TMK	A.TMK
COMPUTE.TMK	C.TMK
PRINT.TMK	P.TMK [2]
UPDATE.TMK	U.TMK

TELLING T/MAKER HOW TO TALK TO YOUR COMPUTER

The file "TMAKER.UTL" contains all the particulars of how T/Maker will talk to your computer. You never mention this name yourself, but you can change this file by giving instructions to a program called T/Modify.

[2] In this case, a PRINT.UTL file should also be renamed P.UTL.

If you are now looking at the prompt of your operating system, you want to type "tmodify" to run the T/Modify program. Within a few seconds, you should see a menu like the one shown below. T/Modify allows you to change many things about how T/Maker behaves.

A>TMODIFY

T/MODIFY VERSION 3.0?

SELECT OPTION TO BE MODIFIED

1. STANDARD TERMINAL CONFIGURATIONS (IE. 5 THRU 8)
2. SPECIAL CHARACTERS USED BY T/MAKER
3. DEFAULT PROCESSING OPTIONS
4. PRINTER OUTPUT CONTROL
5. CONSOLE ROWS AND COLUMNS
6. CONSOLE CONTROL - EXCEPT 7 AND 8
7. CURSOR ADDRESSING
8. REVERSE VIDEO CONTROL
9. EDITING KEYSTROKES
10. PRINTER OUTPUT TECHNIQUE
11. CONSOLE INPUT TECHNIQUE
12. CONSOLE OUTPUT TECHNIQUE
13. SYSTEM PATCHES
99. SAVE MODIFICATIONS AND STOP
999. STOP WITHOUT SAVING THE MODIFICATIONS

ENTER OPTION NUMBER? 1

You tell T/Modify which item you want to change by typing its number followed by a carriage return (i.e. <RET>). In this instance, type "1" and a <RET>. T/Modify will list a number of standard terminals and computers one at a time. When you see yours listed, type "yes" and <RET>; otherwise, just type <RET>.

HAZELTINE 1420	?
HAZELTINE 1500	?
LEAR SIEGLER - ADM31	? YES

After you have typed "yes," the menu will reappear. Now type "99" and <RET> to save the changes which were made by T/Modify to the TMAKER.UTL file.

ENTER OPTION NUMBER? 99

CHANGES SAVED

A>

PROBLEMS AND ANSWERS

YOUR COMPUTER OR TERMINAL WAS NOT LISTED BY T/MODIFY OPTION 1.

You, yourself, will have to specify how T/Maker should control your terminal. At a minimum, you will have to make use of T/Modify options 5, 6, and 7 to get T/Maker running properly. When doing option 6, be sure to start by giving only what is absolutely required. When you are sure T/Maker is working, you can go back and specify the optional information. Do not forget to do it, though, as it may greatly enhance the performance of the T/Maker editor.

SOME OF THE KEYSTROKE COMBINATIONS DON'T DO WHAT THEY SHOULD.

Usually, T/Modify options 5, 6, and 7 have not been specified correctly. It is also possible that your CP/M is intercepting and acting on some of the keystrokes before T/Maker sees them (the Apple II with the SoftCard does this). In this case you will either have to use T/Modify option 11 (if possible) to get around using CP/M routines (not possible with the Apple II) or you will have to use option 9 to change the problematic keystrokes to some which do not conflict with your CP/M.

A good way to start when option 9 is needed is to set all the keystrokes to some common combination such as "Escape Escape." Gradually, you can go back and change a few at a time in the order they are listed to something more convenient. At each stage, you can check that the ones you most recently added do not conflict with ones entered earlier.

THE SCREEN IMAGE IS NOT ALWAYS WHAT IT SHOULD BE.

You have probably found a way to enter characters into the file which T/Maker regards as valid, but which your terminal regards as special control characters. For example, the tilde is treated as a valid character by T/Maker but is considered a control character by a Hazeltine 1500 terminal. Thus, every time T/Maker tries to put the tilde on the screen, the terminal thinks it should do something special. About the best you can do is to use the tilde as one of the editing keystrokes (say, to move the cursor up). That way, you won't be able to enter one while in the editor. While outside the editor, just don't touch the tilde.

The discussion of T/Modify Option 13 also describes one possible source of this problem that does occur with the North Star Advantage and a solution for it.

CUSTOMIZING WITH T/MODIFY

INTRODUCTION

All customizing of T/Maker is done by changing a file named "TMAKER.UTL" ("TMAKER/UTL" for TRSDOS) with a program called T/Modify. To use this program, you must have the files below on your disk.

OPERATING SYSTEM			
CP/M80	CP/M86	MSDOS or PCDOS	TRSDOS
TMAKER.UTL	TMAKER.UTL	TMAKER.UTL	TMAKER/UTL
TMODIFY.COM	TMODIFY.CMD	TMODIFY.EXE	TMODIFY/CMD
TMODIFY.UTL			

If you have a file called TERMINAL.UTL on your distribution diskette, you may also wish to have this file available on the same diskette.

Only TMAKER.UTL of the above is needed to run T/Maker, itself, therefore you can remove them from your working diskette after you have T/Maker properly set up. You can run the T/Modify program by typing "tmodify" in response to the prompt of your operating system.

T/Modify will present a menu of possible changes from which you can select one by typing the corresponding number followed by a <RET>.

When you are finished using T/Modify, you can select option 99 to save the changes you have made or option 999 to disregard the changes you have made. If you attempt to save the changes and get the message "CHANGES NOT SAVED," the typical problem is that your disk is too full to create the modified TMAKER.UTL file. You can erase a file that you don't need from the disk and try again.

All of the options of T/Modify are described in the following sections. To get a feeling for what you change with T/Modify, try all the options, and then finish with option 999. This way, you can see all T/Modify prompts without affecting your current TMAKER.UTL file even if you answer some of the prompts incorrectly.

[1] STANDARD TERMINALS

T/Modify option 1 can initialize options 5, 6, 7, and sometimes 8 on its own for a variety of standard terminals. Even though you may use option 1, it is wise to check the results by using options 5, 6, 7, and 8 and comparing the numbers inserted against the codes given in the manual for your terminal. Make sure that a zero has not been inserted for an optional control feature such as Line Delete. Although T/Maker would work under such circumstances, the ability of the editor to look smart would be degraded. You may also need to adjust the number of nulls inserted by this option (as a timing delay) depending on the communication speed between your computer and terminal.

[2] SPECIAL CHARACTERS USED BY T/MAKER

Option 2 can be used to change some special characters used by T/Maker. For example, the Align and Print commands interpret a ":" in the first column of a line to represent a print design command, if you want to use some other character, you can make the change with this option.

Selecting T/Modify Option 2 will cause T/Modify to display the characters you can change that have special meanings. If you do not want to change a particular character, you simply type a <RET>. If you do want to substitute another character, you should type the desired character. You may also change the role of the comma and period in the representation of numbers, so Europeans may use their conventions.

Under the next two headings, two very special, special characters that can be set with T/Modify are discussed.

The Default Data Drive:

While not really a character in the proper sense, you can set a letter to be used as the drive name whenever you are running T/Maker and ask for a file without specifying a drive name. This is equivalent to using the DATA command which is described in the File Management Section. If you usually run T/Maker with the programs on drive A and your data on drive B, then by all means you should set the default drive for data to B. This means you will no longer have to put the "B:" before your file names. To have no default drive, you can respond to the prompt with a space followed by a <RET>.

T/MAKER will continue to look for all its program files on the currently logged-in drive even if a default data drive is given.

The High Bit Character:

If you specify to T/Modify a nonblank character as the High Bit Character, then whenever that character is typed before another one (except itself), the result will be that the number used to represent the following character under normal circumstances (i.e. 32 to 126, read T/Modify option 9 to understand how characters are represented) will be increased by 128.

In other words, if the high bit character is typed before an "a", the number used to represent the "a" will be 225 instead of the usual 97 (a table in the appendices shows the numbers corresponding to the various characters). For most terminals, this will still appear as the letter "a" on the screen. On a few systems this special "a" will be displayed in reverse video. On a very few systems, the "a" may be displayed as a graphics or other special character.

The point of doing this is that this special "a" is distinct from the normal "a" even though you may not be able to see the difference. Suppose, for example, that you make the high bit character the at-sign (i.e. "@"). And suppose that when option 2 of T/Modify asks for the bold character (i.e. the character used to surround bolded text), you type the "@" and then the "~". You have now made your bold character 128 more than a normal tilde. Under these circumstances, you could use the normal tilde in your files as a regular character. And, if you wanted bolded text, you would have to remember to touch the "@" before the "~" to produce the bold character.

The high bit character can be used to create a whole other character set. It is especially useful for controlling a printer when used in conjunction with a printer translation table (see next subsection).

Another purpose for using the high bit character is to fool T/Maker when for some reason you do not want normal rules to apply. For example, suppose you want three spaces somewhere in a paragraph like these- . Normally, the Align command decides the spacing, but here, three high bit spaces were put instead, so Align would not see them as spaces, but would consider "these- ." as one word. Thus the correct result is achieved.*

* On most printers, high bit spaces will print as regular spaces. Even if they did not, however, a printer translation table could be set up to print a high bit space as a regular space.

[3] CHANGING DEFAULT PROCESSING OPTIONS

Option 3 can be used to change a few default processing conventions. It is convenient to have the defaults correspond to what you typically want. The most significant option is described next.

T/Modify will ask you a number of questions when this option is selected. You should respond with the letter "Y" for "yes" or "N" for "no".

Translating Output To The Printer:

T/Maker can translate a given character into a series of other characters or control codes when printing. This is accomplished by first selecting option 3 and answering "yes" to the question "SHOULD CHARACTERS BE TRANSLATED WHEN PRINTING?"

This question is followed by "WHAT NUMBER SHOULD END A TRANSLATION TABLE LINE?" The meaning of this question will be clear later. The question, itself, may be answered with a number from zero to 255.

The next requirement for using this capability is to develop a translation table. You may use the T/Maker editor to develop such a table and must give it the name PRINT.UTL. It must reside on the same disk as the PRINT.TMK file. If you have specified that translation should be done and do not provide a PRINT.UTL file, you will get the message "PRINT.UTL FILE MISSING" when you attempt to use the Print command. You may use a blank PRINT.UTL file if you want nothing translated for a particular run.

A translation table is a series of lines. The first element on the line gives the character to be translated as a decimal number or an equal sign (i.e. "=") followed by the character in question. The following elements are the characters or codes that should be output when the specified character has been found. They are separated from each other by one or more spaces and may be given as decimal numbers or as an equal sign followed by a character. The list of codes is terminated by a special number which is usually a zero. The number which terminates the list of codes can be set by responding to the question "WHAT NUMBER SHOULD END A TRANSLATION TABLE LINE?" which is mentioned above. The ending number may be followed by a comment.

An example PRINT.UTL file is shown below.

Character Found	Characters Printed	Remarks
=[=Z 8 =N 8 =# 8 == 0	dark block
=]	=[8 =]	0 light block

What this table does is to print a left bracket as a "Z", followed by a backspace (i.e. 8), followed by an "N", followed by a backspace, followed by a "#", followed by a backspace, followed by an equal sign. This would produce a black spot.

A right bracket would be translated when printed as a left bracket, followed by a backspace, followed by a right bracket. This would produce a box type symbol. Such symbols are useful for printing bar charts on word processing printers.

This facility can often be used to control your printer for a variety of purposes ranging from graphics to foreign languages. Normally, it is advisable to translate a character into a series of codes which occupies a single print position just like the character does on the screen. Otherwise, things may not line up on paper as they appear on the screen.

The only problem with translating characters is that you have to use up bonafide characters to create new sequences for your printer. The high bit character facility, described earlier, is extremely useful as a way to get extra unique characters without giving up the standard ones.

Perhaps, solving a particular problem will shed some light on these issues. Consider the case of a NEC Spinwriter printer with the multilingual print thimble. Let's suppose we want to use such a configuration to write a letter in Spanish. To do so, we would need to produce some special characters which are not normally available. We might therefore develop the translation table shown next.

Character Found	Characters Printed	Remarks	To Get, Type
225	=a 8 14 =@ 15 0	a with an accent	HBC* then "a"
229	=e 8 14 =@ 15 0	e with an accent	HBC then "e"
233	=i 8 14 =@ 15 0	i with an accent	HBC then "i"
239	=o 8 14 =@ 15 0	o with an accent	HBC then "o"
245	=u 8 14 =@ 15 0	u with an accent	HBC then "u"
191	14 =L 15 0	upside down ?	HBC then "?"
238	=n 8 =~ 0	small n with tilde	HBC then "n"
206	=N 8 14 =< 15 0	big N with tilde	HBC then "N"

Let's start with an accented "a"; this can be produced on the configuration described by sending out an "a", followed by a backspace, followed by a shift out code (i.e. 14), followed by an "@" (which corresponds to the accent when the shift has been made), followed by a shift in (i.e. 15) to return to unshifted output.

* HBC stands for High Bit Character

Since we need our normal "a" for when an accent is not required, we have chosen to translate the number 225 which is the normal code for an "a", 97, plus 128. This code can be produced by typing the high bit character before the "a". A similar strategy has been used for all the lower case accented letters. Thus, we can produce the accented letters just by typing the high bit character before the lower case vowel to be accented.

If instead of choosing this approach, we had decided to translate the accent as a character on its own into an accent followed by a backspace, this would leave the carriage so that the following letter would be typed under the accent. The flaw in such an approach is that what appears as "'a'" on the screen would only occupy one character when printed. Thus, the result on the printer would be different in terms of alignment than that seen on the screen. Such a discrepancy could create all sorts of problems.

As far as the other translations above are concerned, they are a relatively straightforward implementation of the codes required by the NEC printer and multilingual thimble to produce the various characters.

Below is the translation table used to print this manual on an Epson printer. Notice that 128 is used to end a line instead of the usual zero.

Character Found	Characters Printed	Remarks	To Get, Type
210	27 64	128	Reset
197	14	128	HBC then R
206	20	128	Double Width-On
195	15 32	128	Double Width-Off
227	18 32	128	Condense-On
194	27 69 32	128	Condense-Off
226	27 70 32	128	Bold (emphasize)
175	27 =S 0 32	128	Bold-Off
191	27 =S 1 32	128	Superscript-On
173	27 =T 32	128	Superscript-Off
196	27 =G 32	128	Subscript-On
228	27 =H 32	128	Subscript-Off
213	27 45 1 32	128	Super/Sub-Off
245	27 45 0 32	128	Double Print-On
201	27 52 32	128	Double Print-Off
233	27 53 32	128	Underline-On
211	27 65 =10	128	Underline-Off
			Italics-On
			Italics-Off
			10/72 Line Space
			HBC then S

The rule of translating a character into something which also occupies a single character has not always been obeyed. since of these options tend to be used with a single line of text which is not intended to be justified. One exception is "condensed" which includes a space (i.e. 32) as part of the specification.

[4] PRINTER OUTPUT CONTROL

Option 4 can be used to customize T/Maker for your printer. In most cases, no customization is required. When this option is selected, you will first be asked to give the codes which should be sent to your printer to do a Carriage Return, a Line Feed, a Backspace, and an Advance To Top Of Form. Each of these codes may consist of up to 4 ASCII characters which can be expressed as decimal or hexadecimal numbers. If your printer does not support a backspace, specify all zeros for the backspace.

If the second, third, or fourth character is not needed, a zero should be specified. You may also specify a number of Nulls which should be sent to your printer as a delay after each of these operations has been requested. The number should be between 0 and 255.

If your printer does not have an Advance To Top Of Form, all numbers for the sequence should be given as zero. This will cause T/Modify to prompt for a default physical page size which should be the number of lines on the most standard paper you use. For example, if the paper is 11 inches long and your printer prints 6 lines per inch, then 66 is the correct response. You can override this value by using the ".pagesize" design command when printing. When a physical page size is used, T/Maker will count the lines that it prints and will advance to the next page by skipping an appropriate number of lines.

You can also specify a default length and width of a page with this option. These are the equivalent of the ".length" and ".width" design commands that can be used with Print.

[5] THE NUMBER OF ROWS AND COLUMNS ON THE CONSOLE

Option 5 can be selected to change the numbers of rows and columns on the console. T/Maker systems are normally shipped for a standard 24 rows by 80 columns terminal, however, the system will function in other formats. A few special uses of the screen may look a little off, however, if the width is less than 80 columns.

[6] CONSOLE CONTROL

Options 6 and 7 let you tell T/Maker how to talk to your terminal. The terminal is controlled by sequences of up to four characters (which are expressed as decimal or hex numbers or characters preceded by an equal sign) for the basic editing features (eg. Clear Screen, Line Feed, Carriage Return, etc.). Option 6 will ask you for the correct numbers to send to your terminal for the various results. If the operation requires that less than four numbers or characters be sent, the unneeded numbers should be given as zero. You can also specify a number of nulls (from 0 to 255) to be sent to your console as a delay after using any of the control sequences.

For example, suppose your terminal will clear the screen when it receives an Escape character followed by the letter "G". The correct specification for your terminal would then be 27 for the first number, 71 for the second number, 0 for the third number, and 0 for the fourth number. You might also want to send some number of nulls if you find the computer runs so fast that the terminal is missing some of the characters after a clear screen. The answers for a clear screen could also be given as hex numbers or characters as shown below.

	<u>Decimal Number</u>	<u>Hex Number</u>	<u>Character</u>
First Control Character	27	1BH	=^
Second Control Character	71	47H	=G

The character combination for the Escape is produced by touching the Escape Key on the terminal and not by typing the characters "[" and "^". This technique will not work for the backspace, line feed, or carriage return keys, but it will work for most other nonprintable characters. It is probably best, however, to put such answers in as numbers rather than characters.

Some terminal control functions are optional, but should be given if the terminal supports them. You may, however, want to put them in one at a time after testing T/Maker without the optional control functions. This makes it easier to track down a misspecification. Using all the features of your terminal will cause the editor to do a more efficient job of controlling the screen. If your terminal does not have a beeper, you should consider using the sequence of an exclamation point followed by a backspace in place of beeping. This will leave the cursor in the correct position while providing some minimal feedback that an error has occurred. The appearance of the exclamation point on the screen will not affect the character actually present in your file.

If you want to change a number other than the first in a control sequence, you must re-enter the first number. Otherwise, T/Modify will go directly to the next sequence on the assumption that you did not want to change anything in the sequence.

[7] CURSOR ADDRESSING

Selecting Option 7 permits you to specify for T/Maker the correct means to move the cursor to a precise row and column location for your terminal. First, you must tell T/Maker whether the row or column number should be sent to your terminal as a single character or as character digits. In other words, to send the number 45, should a single character corresponding to 45 be sent or should the two characters "4" and "5" be sent. On most terminals, the number is sent as a single character and not as independent digits.

Next, the number corresponding to the top row is specified as a decimal number. On most terminals it is either 0 or 32 though instances of 1 and 31 have been reported. Next, the number corresponding to the leftmost column is given. On most terminals, it is the same as that used for the top row.

Then, the sequence used to move the cursor is given as a series of numbers or characters preceded by an equal sign (ie. "=Y") terminated by the number zero. Up to eight numbers may be given. This series will be sent to the terminal in the order you give it to move the cursor. To show where the row and column numbers are to be put in the series, the numbers 254 and 255 have special meanings. 254 shows where the column should go and 255 shows where the row should go. The list of numbers should be terminated by a zero unless all eight numbers are used. Finally, an optional number of nulls may be sent as a delay after an instruction to move the cursor. (See pages CN-26 and CN-27 for the correct answers for a few terminals.)

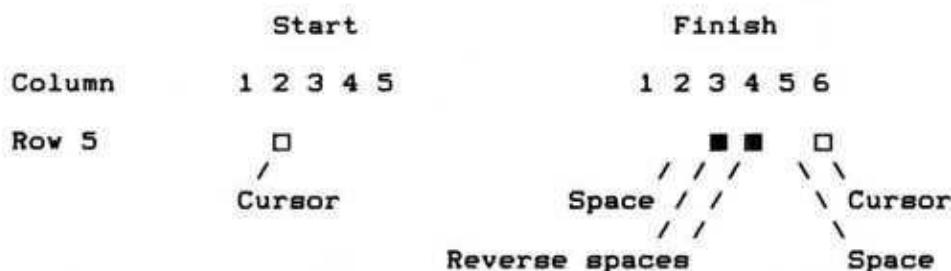
[8] REVERSE VIDEO CONTROL

Selecting Option 8 permits you to specify for T/Maker the correct means to reverse the video for your terminal. Reverse video is optional and is used to draw the frame when in Frame Mode and the bars in bar charts. If it is not specified, a character will be used instead. You may select the character to be used via option 2 of T/Modify.

The use of reverse video is good for advertising, because it looks nice when photographed. On some terminals, it can cause an unpleasant flashing while the screen is being filled. Not that one wants to discourage the use of this option, but if you thought that getting the cursor addressing technique correct was tough, wait till you see what's required for reverse video.

There seem to be three basic techniques for reversing the video on different computer systems. With some serial terminals, a position on the screen is actually used to turn reverse video on or off. In other systems, reverse video is also turned on and off, but no position is used. Finally, in some systems, reverse video is achieved by adding 128 to a normal character or sending a special character; there is no turning on or turning off operation.

What you have to do to use reverse video with T/Maker is first imagine that the cursor is sitting in row 5 column 2 of the screen and then develop a sequence of numbers which when sent to your terminal would result in columns 3 and 4 being reverse blanks with the cursor ending up in column 6 (see diagram below). The sequence should not rely on the fact that you happen to know in which column you started.



Hopefully, the sequence should involve (in the middle) the sending of two identical characters. The first of these identical characters should be replaced by the number 255. The resulting sequence of numbers terminated by a zero is the series of numbers you should give to T/Modify.

Example 1: For an ADM31 Terminal, reverse video must be turned on and off, and the on/off character occupies a display position. The sequence Escape "G" "4" turns reverse video on and advances the cursor one position. The sequence Escape "G" "0" turns reverse video off and also advances the cursor one position. Therefore, to accomplish the problem as posed, the correct sequence to send to the terminal is Escape "G" "4" Blank Blank Escape "G" "0". Coding these values into numbers and replacing the first blank with 255 yields the correct series for T/Modify - 27 71 52 255 32 27 71 48 0.

Example 2: For Radio Shack, reverse video is accomplished by adding 128 to the character to be reversed. So, attacking the problem as posed, a blank must first be sent to advance the cursor from column 2 to column 3; then, the two reversed blanks (128 + 32) can be sent; and finally, another blank must be sent to advance the cursor from position 5 to position 6. The correct sequence for the problem is therefore - 32 160 160 32. Replacing the first 160 by 255 and adding a zero yields the correct series -

32 255 160 32 0.

Example 3: For the North Star Advantage, reverse reverse video is turned on by a Control A (i.e. 1) and turned off by a Control B (i.e. 2). These characters do not however occupy a byte of the display or advance the cursor. Thus, to solve the problem above, the sequence should be - 32 (to advance) 1 (to turn reverse on) 32 32 2 (to turn reverse off) 32 (to advance). Again applying the final rules, the correct sequence for T/Modify is 32 1 255 32 2 32 0.

The role of the number 255 is to note that the next character is the one to repeat to produce multiple reversed blanks. To check your specification, it might help you to know that two consecutive characters on a line should be separated by a space, two reverse blanks, and then another space when a vertical bar is drawn by Frame Mode (see the Edit command) between the two characters.

[9] CHANGING EDITING KEYSTROKES

Option 9 lets you change the keystrokes used to run the editor. You will probably want to leave the keystrokes as they are until you are sure the system is working properly. In any case, you should not select option 9 until option 11 has been set correctly. Usually, option 11 is correct on the initial disk and need not be changed. Some implementations of CP/M, however, intercept a variety of characters before T/Maker has a chance to get at them. For these, you will either have to change the keystrokes to combinations which do not conflict with the CP/M or use direct port input (see option 11, if input is via a port) to avoid having the characters processed by CP/M.

Changing keystrokes can be tricky. It is sometimes advisable to change all keystrokes to something simple like "@ @", and then introduce new keystrokes a few at a time in the order they are called for by T/Modify. This way you can go back at each step to insure that you have not introduced any conflicts.

In order to change the keystrokes confidently, you really need a crash course on terminals. Such a crash course, starts here.

Basically, when you touch the keys of your keyboard, certain signals are sent to your computer. These signals correspond to the numbers 0 through 127. Some keyboards also send out signals in the range from 128 to 255 (see the discussion of the Character Input Mask under T/Modify option 13); and some keys may send out a sequence of signals rather than just one. We won't worry about those for the moment.

The standard characters correspond to the numbers 32 to 126. 127 is usually sent out on most keyboards by a key marked "DELETE" or "RUB OUT." Some of the numbers from 0 to 31 may be sent by particular keys. For example, the Carriage Return Key always sends a 13.

Most keyboards have a key marked "CONTROL" or some abbreviation such as "CTRL." If this key is held while another key is touched, it modifies the signal which is sent out. For example, holding down the Control Key while typing "A" sends out a 1. In fact, the letters "A" to "Z" when used in conjunction with the Control Key send out the signals 1 to 26. The number sent is the same whether the letter is upper or lower case. A carriage return (i.e. 13) is identical to a <CTRL> M.

The Control Key may also be held in conjunction with the key for "@" to send out a zero, as well as the keys for "[", "\", "]", ".", and "_" to send out the numbers for 27 through 31. Holding the Control Key while touching some other character generally makes no sense.

If a terminal has cursor keys, they usually send out the equivalent of some control characters. A backspace key almost always sends out exactly the same signal as <CTRL> H.

<CTRL> I which sends out the signal 27 is generally referred as an "Escape" (i.e. <ESC>), and most keyboards have a key which sends out this signal directly. Some terminals also have one key marked "FUNCTION" or perhaps a series of special function keys. If you have a single function key, then touching that key followed by some character or number will usually cause the terminal to send an opening signal, followed by the signal for the character you typed, and then followed by the signal for a carriage return. On some terminals, only a subset of keys may be touched after the Function Key. On a few terminals, the function keys send out normal characters and are therefore not that useful except when preceded by <ESC> or some other character distinguishable from a normal letter.

From these options, you will want to make a plan of what keys you want to touch to have what done. T/Maker can accept any sequence of up to four signals for any feature. Naturally, if you have cursor keys, a tab key, etc., you will probably want to use those keys for the corresponding editing functions. This eliminates some possibilities. Also, if the second, third, or fourth character in a sequence is a letter, upper and lower case are considered identical (i.e., <ESC> A is the same as <ESC> a).

When using option 9, the program shows you the current keystrokes for each feature. This means you can start by just changing the features for which you have applicable keys, and then seeing what is left over.

There are a few good practices to follow. You will note, for example, that the default keystrokes for "Set left limit" and "Set right limit" for moving columns are <ESC> (and <ESC>). This is easy to remember, and the overhead for having to type two keys in sequence is not that much of a nuisance, since these things are done once and then you are done with them. This is not a feature you would repeat in succession such as "Character Insert" or "Line Delete." There is, however, one nuisance with these sequences. The problem is that "(" and ")" are usually shifted keys. In other words, you have to hold the Shift Key while typing another key. You may wish to change them to an <ESC> 8 and <ESC> 9 if the "(" is over the "8" and the ")" is over the "9". This gives you the best of both worlds - something which is easy to remember and easy to type. If you look at the escape sequences used for the defaults with T/Maker, you will see that the same logic may be applied. Note also how the Escape Key is used to exaggerate the effect of a normal key in the case of cursor movements and the Carriage Return Key. Whereas <CTRL> J may move the cursor down one line, <ESC> followed by <CTRL> J moves the cursor forward one screen.

If a feature is something you might repeat, it is probably wise to use <CTRL> "something" rather than <ESC> "something." This will permit you on most terminals to simply hold down the keys to perform the function over and over again. In these cases where repetition is handy, it is easiest if you avoid keys which send out more than one signal, since by the time T/Maker has processed the first group, it may have missed the first signal of the second group, and begin with the second signal of the second group. (See the discussion of the Console Input Delay Factor under option 13 which can be used to solve this problem.)

Some terminals, such as an ADM31, have keys marked "LINE DELETE," etc. These would obviously be excellent candidates for the corresponding editing features except that on some early models they don't send out any signal at all and do the operation on their own. These keys can obviously lead to a significant discrepancy between what you see on the screen and what T/Maker thinks is there. Clear Keys also have a tendency to clear the screen without sending any signal to the computer. If touching a key four times in a row does not cause T/Modify to prompt for the next set of keystrokes, you can be relatively sure that the terminal sends out no signal for that key.

Finally, avoid generating any ambiguous sequences. If <ESC> ? stands for something on its own then no other sequence should consist of <ESC> ? followed by another character. Obviously, T/Maker can not read the future and guess in advance whether you are going to type that other character or not. Also, do not use <CTRL> @ which is a zero and will generally be ignored outright by T/Maker.

When using T/Modify with this option, the program will run through the editing features one by one displaying the current

keystrokes associated with a particular feature as "W" (meaning <CTRL> W) or "ESC ?", (meaning <ESC> ?), etc.

If you do not want to change a particular one, simply type a blank. Otherwise, type what you want. If you have not used up the allotted four signals, type a blank to cause T/Modify to go on to the next prompt.

The new keystrokes are not displayed when you enter them. If you want to see them, you can simply select option 9 a second time, and T/Modify will show the keystrokes entered the last time as the current keystrokes even though you would still need to select option 99 at the conclusion of T/Modify to make the changes permanent.

If you see strange things going on when the keystrokes you entered are listed, see the discussion of the Character Input Mask under T/Modify Option 13. You may be a victim of the problem described there.

[10] PRINTER OUTPUT TECHNIQUE

Option 10 of T/Modify can be used to select the technique to be used for communicating with your printer. The possibilities are CP/M BIOS calls, CP/M BDOS calls, or direct port I/O. The third option should not be selected unless there is some special reason.

If the third option is selected you will be asked to provide: the status port number, the data port number, a polarity, and a ready flag. The assembly language code which drives the printer based on these values is the following:

```
pra:    push psw      ; print register a
p1:    in  status     ; input from status port
       xri polar      ; Exclusive OR with polarity
       ani ready      ; And Immediate with the ready flag
       jnz p1         ; jump if not ready
       pop psw
       out data       ; output to data port
```

[11] CONSOLE INPUT TECHNIQUE

Option 11 permits you to select the technique used for console input as either CP/M BDOS calls (for CP/M 2.0 or above), CP/M BIOS calls (1.4 or above), or direct port input using the console status and data ports as well as a polarity and ready flag. This last option should not normally be selected, but is sometimes necessary if the BIOS routines have been coded such that direct character input is not really supported. The need for this occurs, for example, on some Vector Graphic machines.

The following table shows the input data port, status port, polarity and ready flag for a few computers.

	Data Port	Status Port	Polarity	Ready Flag
North Star Horizon	2	3	0	2
Vector MZ (Bitstreamer Board)	2	3	0	2
Vector System B (Flashwriter Board)	1	0	0	64

The assembly language code to input a character from the console is the following:

```
    cin:    in    status ; input from console status port
            xri   polar  ; Exclusive OR with the polarity
            ani   ready  ; And Immediate with the ready flag
            jz    cin    ; jump if no character
            in    data   ; input from console data port
            ret
```

If it is necessary to strip polarity from the incoming character by performing an "AND" with 127, this can be accomplished by selecting option 13.

[12] CONSOLE OUTPUT TECHNIQUE

Option 12 permits you to select the technique used for console output as either CP/M BDOS calls (for CP/M 2.0 or above), CP/M BIOS calls (1.4 or above), or direct port output using the console status and data ports as well as a polarity and ready flag. This last option should not normally be selected, but is sometimes necessary if the BIOS routines have been coded such that direct character output is not really supported.

The following table shows the output data port, status port, polarity, and ready flag for a few computers.

	Data Port	Status Port	Polarity	Ready Flag
North Star Horizon	2	3	0	1
Vector MZ (Bitstreamer Board)	2	3	0	1

The assembly language code to output to the console is the same as shown for option 10 except different port numbers may be used.

[13] SYSTEM PATCHES

There are two "system patches" which can be implemented by selecting this option of T/Modify. They both concern the inputting of characters from the console.

Console Input Delay Factor:

The first variable you can specify is called the "console input delay factor." It should be a number between zero and 255. The factor controls the strategy used by the T/Maker editor to determine where one set of keystrokes starts and ends. In a system with buffered input where you would like to have type-ahead, the correct number would be zero. For other systems, most anything from 1 to 255 will work. The delay factor can solve the following problem:

You have a key on your keyboard which sends out more than 1 true character, for example, <ESC> A. When you hold this key down, <ESC> A is sent in rapid succession. There is the possibility that T/Maker will get back too late from processing the first <ESC> A to pick up the next sequence correctly. It may see the "A" but may have missed the <ESC>. The result would be an "A" appearing on the screen. A proper adjustment of the delay factor will solve this problem.

Console Input Mask:

The second variable you can specify is called the "console input mask." It permits you to "AND" (as in Assembler) the incoming data from the console with a mask. You specify the mask. Normally, the value is set to 255 so that no masking is done by T/Maker, although the CP/M calls may do their own masking. Here is a problem where this feature is useful.

On the North Star Advantage, the cursor key pad sends out characters with values in excess of 127. T/Maker regards these as standard characters and echos them back to the console. On some machines, values in this range are used as valid graphics characters. The console, however, regards them as control characters and does a variety of operations on the screen. The result is total confusion between T/Maker and the Advantage if these keys are touched.

By selecting a character input mask of 127, T/Maker will see these keys as sending out control characters and will therefore not accept them unless they are used as keystrokes for the various editing features.

Closing All Files:

For 16-bit operating systems, an extra option exists. This option can be used to force T/Maker to close all files after reading them. If you see your system generating a message such as "TOO MANY FILES OPEN" or "TOO MANY FILES ACTIVE" then you should select Option 13 of T/Modify and answer the third question you will be asked with "yes."

ASCII CODES AND CHARACTERS

<u>Character</u>	<u>Decimal Number</u>	<u>Character</u>	<u>Decimal Number</u>	<u>Character</u>	<u>Decimal Number</u>
<CTRL> @	0	,	44	X	88
<CTRL> A	1	-	45	Y	89
<CTRL> B	2	.	46	Z	90
<CTRL> C	3	/	47	[91
<CTRL> D	4	0	48	\	92
<CTRL> E	5	1	49]	93
<CTRL> F	6	2	50	.	94
<CTRL> G	7	3	51	-	95
<CTRL> H	8	4	52		96
<CTRL> I	9	5	53	a	97
<CTRL> J	10	6	54	b	98
<CTRL> K	11	7	55	c	99
<CTRL> L	12	8	56	d	100
<CTRL> M	13	9	57	e	101
<CTRL> N	14	:	58	f	102
<CTRL> O	15	;	59	g	103
<CTRL> P	16	<	60	h	104
<CTRL> Q	17	=	61	i	105
<CTRL> R	18	>	62	j	106
<CTRL> S	19	?	63	k	107
<CTRL> T	20	@	64	l	108
<CTRL> U	21	A	65	m	109
<CTRL> V	22	B	66	n	110
<CTRL> W	23	C	67	o	111
<CTRL> X	24	D	68	p	112
<CTRL> Y	25	E	69	q	113
<CTRL> Z	26	F	70	r	114
<ESC>	27	G	71	s	115
<CTRL> \	28	H	72	t	116
<CTRL>]	29	I	73	u	117
<CTRL> .	30	J	74	v	118
<CTRL> -	31	K	75	w	119
Space	32	L	76	x	120
!	33	M	77	y	121
"	34	N	78	z	122
#	35	O	79	{	123
\$	36	P	80		124
%	37	Q	81	}	125
&	38	R	82	~	126
,	39	S	83	DEL	127
(40	T	84		
)	41	U	85		
*	42	V	86		
+	43	W	87		

TERMINAL CONTROL SEQUENCES

ANSII STANDARD

ADDRESSING THE CURSOR

Character Digits?..... Yes

Top Row..... 1
Leftmost Column..... 1

Number 1 27
Number 2 91
Number 3 255
Number 4 59
Number 5 254
Number 6 72
Number 7 0
Number 8

MANDATORY FEATURES

Carriage Return.....	13	0	0	0
Line Feed [1].....	10	0	0	0
Backspace.....	8	0	0	0
Clear Screen.....	27	91	50	74

OPTIONAL FEATURES

Make Beep [2].....	7	0	0	0
Erase to End of Line...	27	91	48	75
Erase to End of Screen.	27	91	48	74
Line Insert.....	27	91	48	76
Line Delete.....	27	91	48	77
Character Insert.....	27	91	48	64
Character Delete.....	27	91	48	80

- [1] Some terminals (eg. Hazeltine 1500) have both a Line Feed and a Cursor Down. Be sure to use the one (usually the Line Feed) which scrolls automatically on the bottom line.
- [2] If you do not have a beeper, consider using an exclamation point followed by a backspace (ie. CURSOR LEFT) so that you will get some visual feedback for an error. On most systems, the sequence for this would be - 33 8.
- [3] Not supported by the TI Professional. Use all zeros instead.
- [4] Not supported by the DEC Rainbow. Use all zeros instead.

	<u>ADM31</u>	<u>ADM3A</u>	<u>HAZELTINE 1420 [5]</u>	<u>SOROC 1Q120</u>
ADDRESSING THE CURSOR				
Character Digits?.....	No	No	No	No
Top Row.....	32	32	0	32
Leftmost Column.....	32	32	0	32
Number 1	27	27	27	27
Number 2	61	61	17	61
Number 3	255	255	254	255
Number 4	254	254	255	254
Number 5	0	0	0	0
Number 6				
Number 7				
Number 8				
MANDATORY FEATURES				
Carriage Return.....	13 0	13 0	13 0	13 0
Line Feed.....	10 0	10 0	10 0	10 0
Clear Screen.....	27 58	26 0	27 28	27 42
Backspace.....	8 0	8 0	8 0	8 0
OPTIONAL FEATURES				
Make Beep.....	7 0	7 0	7 0	7 0
Erase to End of Line...	27 84	0 0	27 15	27 84
Erase to End of Screen.	27 121	0 0	27 24	27 89
Line Insert.....	27 69	0 0	27 26	0 0
Line Delete.....	27 82	0 0	27 19	0 0
Character Insert.....	27 81	0 0	0 0	0 0
Character Delete.....	27 87	0 0	0 0	0 0

[5] 126 may be used in place of 27 depending on a console switch.

INDEX

absolute value	ES-6
absolute changes between columns of numbers	ES-46
accuracy of calculations	ES-52 to 54
advancing the printer to a specific line	WP-80
ALIGN command	WP-80
aligning shortcuts	WP-35 to 34
aligning wedges	WP-79
spacing within aligned text	WP-79
AMEND command	SC-21
ambiguous word reference	SC-17
arctangent of numbers	ES-6
ARRANGE command	LP-1
ASCII files	DS-11
average of numbers	ES-6
BAR command (for bar charts)	GR-1
batching commands	DS-1
block text (aligning wedges)	WP-80
blocking text to appear on one page	WP-47 to 47
bold toggle character, changing it	CN-8
bolded text	WP-59 to 59
Bottom Routines	WP-51 to 54
buffer, the	WP-79
Calculation Lines (see Horizontal Calc. Lines)	
Calculation Strips (see Vertical Calc. Strips)	
carriage return modes	WP-80
cash flow examples	ES-76, ES-86
centering text (aligning wedges)	WP-79
changes between columns of numbers	ES-46
"changed" field	DB-8
as a general conditional	DB-10, DB-11
checkbook examples	ES-61, ES-79
.clean (print design command)	WP-66, WP-70, ES-95
CLEAN command	ES-99
CLIP command	FM-6
cluewords	DE-2
column percents	ES-48
COMBINE command (combining tables)	ES-96
comments, in files but not printed	WP-79
comparing files	LP-4 to 5
COMPUTE command	ES-1
COMPLETE command	DB-11 to DB-13
Constant Calculation Line	ES-38 to 43

.continue (print design command)	WP-61 to 61
examples of	WP-80
<continue> option	DB-12
Control Codes (for Calculation Lines)	ES-27 to 34
method of checking them	ES-70 to 72
Control Key	WP-3
copying lines	WP-80
CORRECT command	SC-11
correlation coefficient, example calculation	ES-89
cosine of numbers	ES-6
CREATE command	FM-2
crossword puzzles	SC-19
cumulative values	ES-45, ES-58
cursor movements	WP-4 to 6
cursor movement	DB-5 to DB-8
finding specific records	DB-7
from record to record	DB-6
inserting and deleting records	DB-6
moving or duplicating records	DB-7
revising a record	DB-6
within a record	DB-5
customizing T/Maker	CN Section
 DATA command	FM-7, CN-8 to 9
data-base files as tables	DB-10, DB-11
data-base, definition of	DB-3
data encryption	DE-1
dates, calculating with	ES-78
debugging tables	ES-70 to 72
Delay (option to Print command)	WP-79
DELETE command	FM-4
depreciation (example table)	ES-80
Design Commands	WP-79
in tables	ES-65
dictionaries, building	SC-1, 22
dictionaries, creating	SC-10, 22
dictionaries, additional	SC-5
dictionaries, modifying	SC-21
dictionary arithmetic	SC-25
DIF files	DS-11
disk file names	FM-1
DO command	DS-1
double spacing	WP-80
DROP command	LP-3
duplicating lines	WP-80
 EDIT command	WP-1
editing keystrokes, changing them	CN-17
electronic spreadsheet	ES Section
end designation <end>	DB-4
entering numbers in a table	ES-67 to 68
entering the editor	WP-2
errors - for compute command	ES-73

Escape Key	WP-3
Example Line	ES-10
changing it within a table	ES-66 to 67
invisible columns	ES-65 to 66
zones for numbers	ES-67 to 68
zones for symbols	ES-24 to 25
exponential function	ES-6
Fetch and Store (restricted calculations)	ES-55 to 60
field names and lengths	DB-3
importance of case	DB-9
long names in short fields (#)	DB-7
picking them	DB-8
field, definition	DB-3
justification	DB-7, DB-8
multiple fields on a line	DB-7
file management	FM Section
file names	FM-1
files, discussion of	IN-3
FILES command	FM-6
finding strings	
inside the editor	WP-14 to 13
outside the editor	LP-2
footnotes	WP-80
forced new page	WP-79
foreign languages	CN-10 to 13
number conventions	CN-8
form definition	DB-4, DB-5
dressing it up	DB-2
maximum size	DB-2
multiple form definitions	DB-3
switching from one to another	DB-3
variations and refinements	DB-2
form letters, example of	WP-80
fractions in a table, example of	ES-77
Frame Mode	WP-26 to 25
frequency distributions	LP-11
From page-number (option to Print command)	WP-79
General Calculations	ES-21
GET command	FM-3
grammar school arithmetic	ES-1
graphics	GR Section
graphics characters	CN-10 to 13
GROUP command	DB-14 to DB-2
counting records	DB-15
"list" option	DB-15
report of	DB-16, DB-2
group totals from a list	LP-11
growth rates	ES-48 to 49, ES-81
hand calculator problems	ES-74
headings (repeated on pages)	WP-51 to 54

here option <here>	DB-9, DB-10, DB-8
HELP command	DS-4
High Bit Character	CN-9
Horizontal Calculation Line	ES-21 to 23
placing symbols	ES-24 to 27
Control Codes	ES-27 to 34
Sequence Numbers	ES-35
.indent (print design command)	WP-80
indented paragraphs (aligning wedges)	WP-80
INFO command	FM-10, WP-79
information keystrokes	WP-23
INSERT command	FM-5
Insert Mode	WP-79
inside the editor	WP-2
interfacing with other programs	DS-11
invisible columns in a table	ES-65 to 66
invisible rows in a table	ES-70
iteration (with Compute command)	ES-63 to 64
KEEP command	LP-3
Keystroke Macro	WP-18 to 18
Kill character	FM-12
labels, examples of	WP-63, WP-80
.length (print design command)	WP-79
letter, example of	WP-79
limits - for Compute command	ES-73, ES-83
(Line-number/Position-number) command	FM-10
LIST command	FM-6
list processing	LP Section
listing file-names on a disk	FM-6
LOAD command	TU-1
logarithms of numbers	ES-6
long document, example of	WP-71 to 70
long documents, linking their files	WP-61 to 61
mask files (for Load and Unload commands)	TU-4 to 5, TU-8
MATCH command (matching files)	LP-4 to 5
mathematical operators	ES-2
maximums of numbers	ES-7, ES-39, ES-88
menus, creating custom	DS-4
MERGE command	FM-5
merging tables	ES-99
minimums of numbers	ES-7, ES-39, ES-88
MIX command	DE-1
model numbers (see example line)	
moving a column of numbers to the right or left	ES-47
moving columns	
inside the editor	WP-24 to 23
outside the editor	LP-1
moving lines	WP-80
moving the cursor	WP-4 to 6

moving the screen	
inside the editor	WP-4 to 6
outside the editor	LP-2, FM-10
net present value	ES-51
.newpage design command	WP-79
Non-Aligning Line	WP-79
non-lines (not printed)	WP-79
Nonstop (option to Print command)	WP-79
NOTABS command	FM-11
numbered pages	WP-56 to 55
.number (print design command)	WP-79
number (option to print command)	WP-79
numbered paragraphs (aligning wedges)	WP-80
numbers in a table	ES-67 to 68
optional new page	WP-79
ORDER command	DB-12, DB-13
using SET for frequent ordering	DB-13
order among calculations	ES-36 to 37
order within a General Calculation	ES-3
organizing long documents	WP-61 to 61
overprinting (one line printed over another)	WP-79
page breaks	
automatic	WP-80
forced and optional	WP-79
page length	WP-79
page numbers	WP-56 to 55
page width	WP-80
page-endings	WP-51 to 54
page-headings	WP-51 to 54
.pagesize (print design command)	WP-64 to 63
panelling (for printing wide documents)	WP-80
paragraph separation (for aligned text)	WP-79
pas (restricted calculation)	ES-60 to 62
passwords	DE-1
percentage changes	ES-46, ES-82
percents (columns)	ES-48
percents (converting whole tables)	ES-42 to 43
percents (rows)	ES-64
percents (useful terminators- % # sfh sfo)	ES-6
physical page (as printed)	WP-64 to 64
placing numbers in a table	ES-67 to 68
placing symbols (on Calculations Lines)	ES-24 to 27
powers of numbers	ES-7
PRINT command	WP-80
options of the Print command	WP-42 to 41
Print Design Commands	WP-79
Print Design Commands (in tables)	ES-65
print the working file	WP-79
Printer Translation Table	CN-10
Printer's Prompt	WP-38, WP-79

printing to the screen	WP-79
problems in getting T/Maker to run	CN-6
projections	ES-50, ES-81
prompts - Other	IN-6
Printer's Prompt	WP-38, WP-79
Universal Prompt	IN-5, IN-8
PROOF command	SC-5
quitting the editor	WP-3
Recalled wedges (for aligning)	WP-35 to 34
record definition	DB-3
handling missing data	DB-6
multiple record definitions	DB-12 to DB-15
variation and refinements	DB-6
record keeping, example of	ES-75
redrawing the screen	WP-79
RENAME command	FM-5
replacing strings	WP-14 to 13
inside the editor	LP-2
outside the editor (Replace command)	FM-9
RESET command	ES-37 to 62
Restricted Calculations	ES-38 to 43
Constant Calculation Line	ES-44 to 51
Special Notations	ES-55 to 60
Store and Fetch	ES-60 to 62
Passing Values	ES-52 to 54
Rounding	CN-15
reverse video	WP-9 to 11
revising text keystrokes	ES-52 to 54
rounding (Restricted Calculation)	DB-26
rules definition	DB-34
applying a conditional	DB-31
comparison operators	DB-30
conditional clause operators	DB-27
model numbers within	DB-28
presenting zero values	DB-26
requesting mathematical calculations	DB-30
selectively applying rules	DB-34
sequence of processing operators	DB-28
sequentially numbering records	IN-4
running T/Maker	FM-2
SAVE command	WP-4 to 6
screen movements	LP-2, FM-10
inside the editor	WP-79
outside the editor	WP-14 to 13
screen printing	LP-2
searching	DE-1
inside the editor	
outside the editor	
security	

SELECT command	DB-43 to DB-50
automating the procedure	DB-48
conditional selections of records	DB-45
<continue> option	DB-50
from-to options	DB-44
"it" option	DB-49
"list" option	DB-44
used to reformat fields	DB-45
with subsets of records	DB-43
selecting from a list	LP-3
Sequence Numbers (for Calculation Lines)	ES-35
SET command	DB-25 to DB-36
assigning a character string	DB-29
concatenating strings	DB-30
manipulating alphabetic characters	DB-29
shifting text when printing	WP-80
sine of numbers	ES-6
single spacing	WP-80
size of printed page	WP-64 to 64
SORT command	LP-6
sorting lines, a few manually	WP-80
spacing (single or double)	WP-80
spacing (within aligned text)	WP-79
Special Notations (Restricted Calculation)	ES-44 to 51
Special Terminator Symbols	ES-7
SPELL command	SC-17
square root of numbers	ES-6
squeeze justification symbol (!)	DB-8
standard deviation, example calculation	ES-89
start text on line "x"	WP-80
statistics, example calculation	ES-89
STOP command	FM-12
stopping aligning	WP-33 to 32
stopping a command in progress	FM-12
Store and Fetch (Restricted Calculations)	ES-55 to 60
strings	WP-79
Strips (see Vertical Calc. Strips)	
table lookup, simulation of	ES-91
tables within text, example of	WP-79
tabs (saving tab stops with the file)	FM-11
tabs (using them while editing)	WP-80
TABS command	FM-11
Tagged wedges (for aligning)	WP-35 to 34
TALLY command	LP-11
tangent of numbers	ES-6
Terminator Symbols	ES-5 to ES-7
To page-number (option of the Print command)	WP-79
toggles, for bold and underscoring	WP-59 to 58
Top Routines	WP-51 to 54
TRANSFER command	TU-10

unattended processing	DS-1
underscored text	WP-59 to 59
underscore toggle character, changing it	CN-8
Universal Prompt	IN-5, IN-8
UNLOAD command	TU-8
UPDATE command	DB-19 to DB-24
keystrokes	DB-20 to DB-23
Variance, example calculation of	ES-89
Vertical Calculation Strips	ES-8 to 18, ES-69 to 70
entering them	ES-69
invisible rows	ES-70
WAIT command	FM-12
wedges (aligning wedges)	WP-79
.wedges (print design command)	WP-80
wide pages, printing of	WP-80
widowing lines	WP-80
.width (print design command)	WP-80
word processing	WP Section
working area	IN-7, IN-8
working file	IN-3, IN-8
Zero Values Line	ES-19 to 20

TIMakerTM

QUICK REFERENCE

useful terms

Working File—refers to all lines in the file

Position 1



Position 2 etc.

Line 1

Line 2

etc.

Working Area

Refers to all lines from the top line on the screen forward—the area normally affected by a command

Top line on the screen

Top line in the frame
(when in Frame Mode)

15/55 WHAT NEXT? sort n 1 10

Parameters or Options
(some commands require them)

Command name

Universal Prompt

Leftmost position shown
on the screen (or in
the frame)

Line number of the top
line of the screen (or in
the frame)



edit

To enter the editor:

WHAT NEXT? edit or e

To quit the editor: F1

When the editor is quit, the line on which the cursor rests becomes the top line on the screen (or the top line inside the frame, when in Frame Mode).

Revise Text

	MINE	
Insert character	Ins	_____
Insert line	F9	_____
Delete character	Del	_____
Delete line	F10	_____
Delete spaces to next character	Esc Ins	_____
Destructive backspace	←	_____
Erase to end of line	Esc Del	_____
Join lines	Esc F10	_____
Break lines	Esc F9	_____

Move Lines

Clear the buffer	Esc F6	_____
Copy to the buffer	F7	_____
Move to the buffer	F8	_____
Insert the buffer	Esc F7	_____

Search & Replace

Search for string	F5	_____
Continue search	F6	_____
Replace strings	Esc F5	_____

Change Modes

Carriage return mode	Esc R	_____
Insert mode	Esc I	_____
Frame mode	Esc F	_____

Move Columns

	MINE	
Set the left limit	Esc (_____
Set the right limit	Esc)	_____
Copy columns	Ctrl F	_____
Move columns	Ctrl G	_____

Tabs

Tab	← →	_____
Set tab	Esc ← →	_____
Clear tab	Esc Esc ← →	_____
Set all tabs	Esc S ← →	_____
Clear all tabs	Esc C ← →	_____

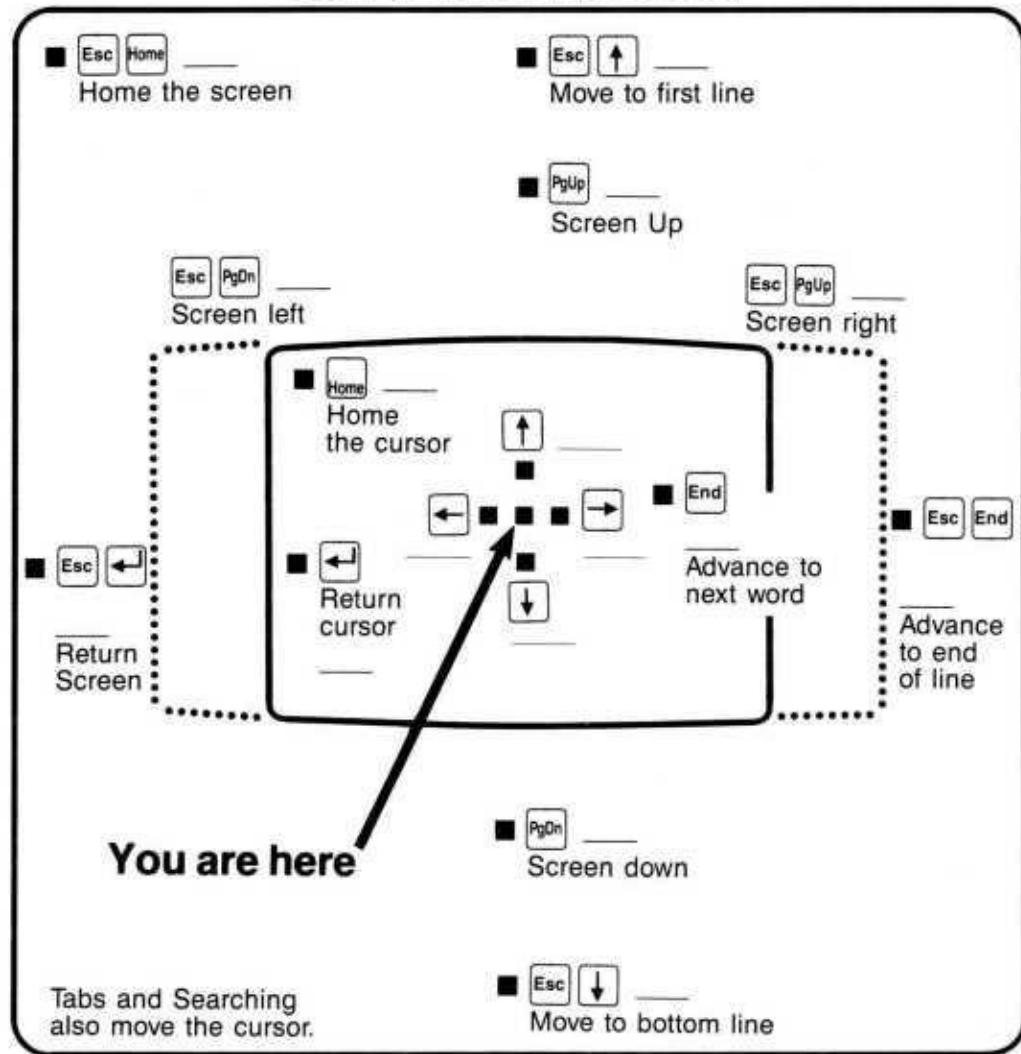
Keystroke Macro

Begin macro	F3	_____
Home macro	Esc F3	_____
End macro or execute once	F4	_____
End macro or execute many times	Esc F4	_____

Others

Quit the editor	Esc F1	_____
Repeat last keystrokes	Ctrl K	_____
Information	F2	_____
Redraw the screen	Esc F2	_____

How to Move the Cursor



How to Use the Keys

■ Esc Z — touch Escape Key and then other key

■ Ctrl Z — hold down Control Key and touch letter shown

■ Esc — KILL CHARACTER
To stop commands in progress

All keystrokes for editing commands may be changed to the user's own preference. See "Customization Notes" in the Reference Manual.

edit

To enter the editor:

WHAT NEXT? **edit** or **e**

To quit the editor: **CTRL Q**

When the editor is quit, the line on which the cursor rests becomes the top line on the screen (or the top line inside the frame, when in Frame Mode).

Revise Text

Insert character	CTRL W	_____
Insert line	CTRL R	_____
Delete character	CTRL E	_____
Delete line	CTRL T	_____
Delete spaces to next character	CTRL U	_____
Destructive backspace	DEL	_____
Erase to end of line	CTRL Y	_____
Join lines	ESC <	_____
Break lines	ESC >	_____

Move Lines

Clear the buffer	CTRL C	_____
Copy to the buffer	CTRL F	_____
Move to the buffer	CTRL G	_____
Insert the buffer	CTRL D	_____

Search and Replace

Search for string	ESC *	_____
Continue search	ESC &	_____
Replace strings	ESC "	_____

Change Modes

Carriage return mode	ESC R	_____
Insert mode	ESC I	_____
Frame mode	ESC F	_____

Move Columns

Set the left limit	ESC (_____
Set the right limit	ESC)	_____
Copy columns	CTRL S	_____
Move columns	CTRL V	_____

Tabs

Tab	TAB	_____
Set tab	ESC TAB	_____
Clear tab	ESC ESC TAB	_____
Set all tabs	ESC S TAB	_____
Clear all tabs	ESC C TAB	_____

Keystroke Macro

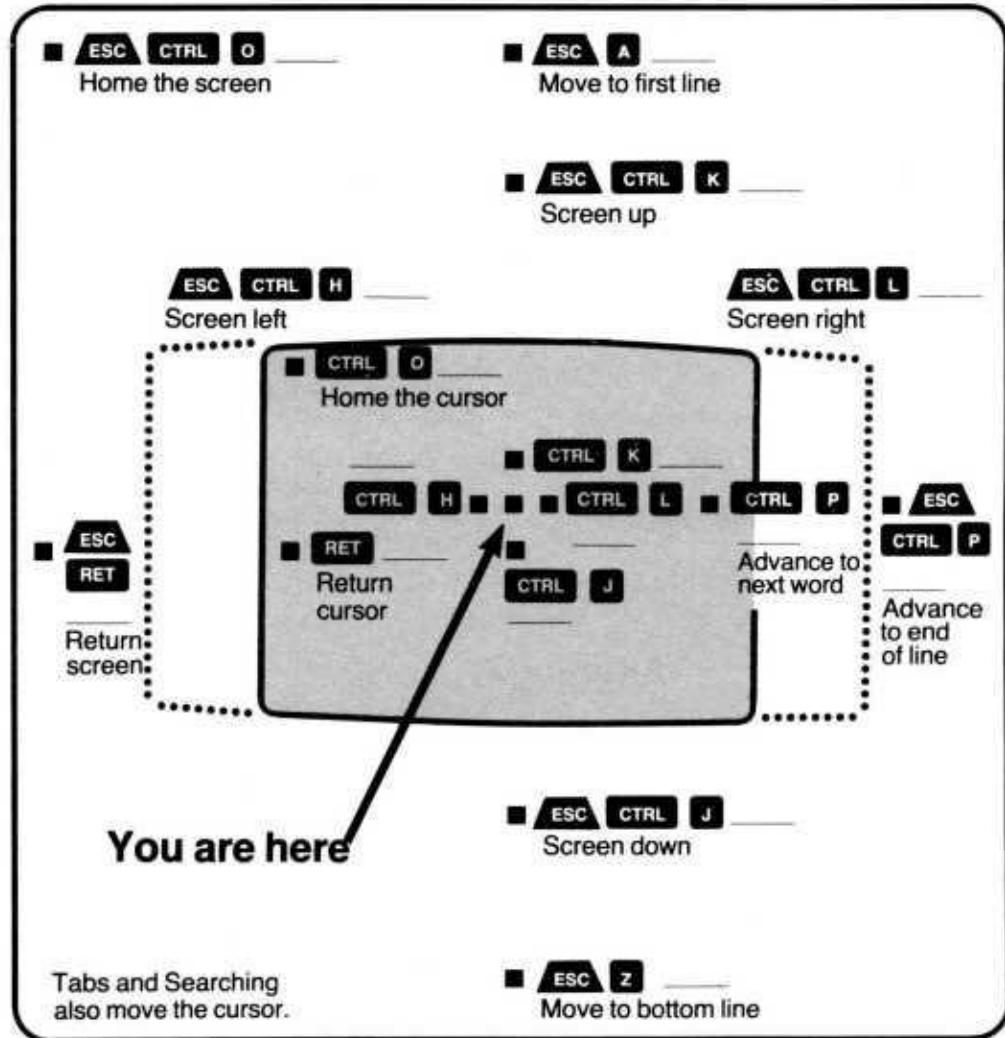
Begin macro	CTRL B	_____
Home macro	CTRL N	_____
End macro or execute once	CTRL X	_____
End macro or execute many times	CTRL Z	_____

Others

Quit the editor	CTRL Q	_____
Repeat last keystrokes	CTRL A	_____
Information	ESC ?	_____
Redraw the screen	ESC #	_____

(See "Addenda" for
IBM Users' Edit pages)

How to Move the Cursor



How to Use the Keys

RET	<i>touch Return Key</i>
ESC	<i>touch Escape Key</i>
CTRL X	<i>hold down Control Key and touch letter shown</i>
TAB	<i>touch Tab Key or CTRL I if the terminal has no Tab key</i>

DEL *touch Delete Key (sometimes marked "Rub")*

All keystrokes for editing commands may be changed to the user's own preference. See "Customization Notes" in the Reference Manual.

align

To align the text in the working area:

WHAT NEXT? **align**

The text will be aligned according to aligning wedges placed in the file. Each set of wedges applies to the text that falls after it in the file until a new set of wedges appears.

To change the margins on one side, specify only the wedges on that side.

Paragraph Separation

Paragraphs must be separated by one of the following:

- a blank line
- a line containing aligning wedges
- a line with a period in position one (i.e., a "design command" line)
- a line with three or more dashes in succession
- a line with three or more underscores in succession

Tagged and Recalled Wedges

A pair of aligning wedges may be "tagged" by a one-letter name.

<< **a** >>

Tagged in this way, the wedges may be recalled for use later on in the file by inscribing a left-side pair of wedges, an equal sign, and the one-letter name.

<< = a

Aligning Options

Unaligned Text

>> <<
All work
and no play
makes
Jack a dull boy

Block Text

<<< >>
All work and no play makes
Jack a dull boy. All work and
no play makes Jack a dull
boy. All work and no play
makes Jack a dull boy.

<<< ->>
All work and no play makes
Jack a dull boy. All work and
no play makes Jack a dull
boy. All work and no play
makes Jack a dull boy.

JUSTIFIED

RIGHT-RAGGED

Centered Text

<<- >>
All work
and no play
makes
Jack a dull boy

Indented or Numbered Paragraphs (First Line Free)

<< >>
All work and no play
makes Jack a dull boy. All
work and no play makes
Jack a dull boy. All work and
no play makes Jack a dull
boy.

<< ->>
1. All work and no play
makes Jack a dull boy.
All work and no play
makes Jack a dull boy.
All work and no play
makes Jack a dull boy.

JUSTIFIED

RIGHT-RAGGED

print

To print the working area or a file on disk:

WHAT NEXT? **print** [options, if any]
[it or disk-file-name]

Printing in progress can always be terminated by typing **CTRL Q** or **ESC**

Options

Options, when used, are interposed between the word "print" and the designation of the file to be printed.

- nonstop** Pre-answers Printer's Prompts with **y**.
- number 5** The first page in the file is assumed to be page number 5.
- from 7** Prints from page 7.
- to 10** Prints through page 10.
- delay 9** Sets the speed of printing to the screen.

Bold and Underscored Text

A string of text can be made bold by preceding and following it with a vertical bar character (|).

A string of text can be underscored by preceding and following it with a back-slash character (\).

The characters used to underscore or make text bold occupy a character space but are not printed.

Note: See Design Commands for other ways to bold or underscore a complete line.

Note: Characters for bold and underscoring may be changed to the user's own preference.

See "Customization Notes" in the Reference Manual.

The Printer's Prompt

The Print command brings the "Printer's Prompt" to the screen.

NEXT PAGE 1 (YES.SCREEN.NO.GO.QUIT)?

One of the following answers should be given:

Answer	Result
y	Prints the page on paper.
s	Prints the page on the screen. The printing may be stopped temporarily (and re-started) by touching the spacebar.
n	Skips the page. The Printer's Prompt appears for the next page.
g	Pre-answers all following Printer's Prompts with y.
o	Terminates printing session.

Design Commands

Design Commands carry out a number of editing functions. They are inscribed into the file but are not printed.

Design Commands always begin with a period in position one. Keywords in Design Commands (e.g., **single**) may be abbreviated to their first three letters (**sin**).

Note: See following pages for descriptions of Design Commands.

print

Design Commands

.width 80	Sets page or panel width (e.g., 80 positions).
.pagesize 66	Sets physical size of a page (e.g., 66 lines).
.length 60	Sets usable length of a page (e.g., 60 lines).
.number 3	Sets page number of current page (e.g., 3). Note: <i>Number 0</i> sets page number of current page and following pages to zero until reset by another .number command.
.top 5	"Top Routine"—a series of lines to be printed single-spaced at the top of each page starting with the page indicated (e.g., page 5). Only the following Design Commands may appear within a Top Routine: .+ .- .= .&
.end	Number signs (#) within Top Routines provide automatic page numbering.
.block	Defines a block of lines to be printed on a single page. Any of the following Design Commands will end the block: .end .block .top .bottom .length .pagesize .newpage .footnote .end
.footnote	Defines a group of lines to be printed single-spaced at bottom of this page or next. Footnotes are placed above Bottom Routines. Only the following Design Commands may appear within a Footnote: .+ .- .= .&
.bottom 2	"Bottom Routine"—a series of lines to be printed single-spaced at the bottom of each page starting with the page indicated (e.g., page 2). Only the following Design Commands may appear within a Bottom Routine: .+ .- .= .&
.end	Number signs (#) within Bottom Routines provide automatic page numbering.

print

Design Commands

. single	Single spaces the following text.
. double	Double spaces the following text.
. wedges	Prints aligning wedges.
. nowedges	Terminates printing of wedges.
. clean	"Cleans" the following table before printing.
. noclean	Prints the following table as is.
. indent 6	Shifts the following text 6 positions to the right.
. indent 0	Prints the following text without indenting.
. +	Bolds all previous line.
. -	Underscores all previous line.
. =	Underscores all words in previous line.
. &	Overprints previous line with next line.
.	The line is ignored (paragraph separator).
. . Hi Jack!	Comment Line—text appears in file but does not print.
. ,	Nonaligning Line—text printed unaligned, minus dot and comma.
. 23	Advances printer to the 23rd line of the page.
. newpage	Advances printer to a new page.
. newpage 5	Advances printer to a new page if less than 5 lines are left on the current page.
. continue xxx	Printing automatically proceeds to disk file named (e.g., xxx). The working file can contain numerous .continue commands. A file printed from disk can contain only one, and that must be on its bottom line.

files management commands

Function	WHAT NEXT?	Notes
clip	<i>clip before or after</i>	"Before" and "after" may be abbreviated to their first letters.
create	<i>create file-name</i>	
To create a working file with a given name.		
data	<i>data drive-name (for example, a or b)</i>	
To establish a default disk drive for all T/Maker text files. Thereafter, any file name lacking a specified disk drive will be assumed to refer to the default drive. To return to no default drive, give a space as the drive name.	<i>data</i>	
delete	<i>delete disk-file-name</i>	
To delete a file on disk.		
do	<i>do</i>	This line is automatically deleted from the working file.
To process the top line on the screen (or in the frame) as if it were typed in after the Universal Prompt.		
files	<i>files drive-name</i>	The listing may be interrupted and restarted by touching the space bar.
To inventory the files on a disk drive.		
get	<i>get or g disk-file-name</i>	
To make a file on disk the working file.		
info	<i>info</i>	
For information about the working file.		
insert	<i>insert disk-file-name</i>	
To insert a file on disk at the top of the working area.		
[line-number]		Examples: 25
[/position-number]		and 35/40
To move the upper left-hand corner of the screen (or of the inside of the frame) without entering the Editor, type the desired coordinates of the new location. Only one coordinate need be specified if the other is not to be changed.		and/150

files management commands

Function	WHAT NEXT?	Notes
list To view a file on disk without affecting the working file.	list <i>disk-file-name</i>	The listing can be interrupted and restarted by touching the spacebar. (See <i>Print</i> for printing on the screen.)
merge To superimpose a file on disk on top of the working area.	merge <i>disk-file-name</i> <i>position</i>	The position parameter defines the start of the horizontal zone of the working area to be overlaid.
notabs (see tabs) To stop subsequent Save commands from saving tabs.	notabs	
rename To give a new (and additional) name to the working file.	rename <i>new-file-name</i>	
reset To prevent a BDOS Read Only (R/O) Error, when attempting to alter a newly inserted disk, type after inserting the new disk.	reset	
save To store the working file on disk.	save or s	This command will save tab stops with the file if the <i>Tabs</i> command was previously invoked. If a previous version of the named file is on disk, that version is renamed <i>file-name.BAK</i> .
stop To stop T/Maker and return to the operating system. To interrupt most major functions in progress.	stop CRTL Q or ESC	
tabs (see notabs) To save tabs with subsequent Save commands.	tabs	
wait To suspend a series of commands until a key is touched at the console, add to series.	wait	

error messages

(General)

General Errors

Most error messages in T/Maker are self-explanatory. Below, a few common problems are described.

OUT OF SPACE

The working file is getting too big. Enter the editor and delete whatever you can or restructure the file into smaller pieces.

DISK FULL

The Save command was used and the disk or disk directory is full. Use the *Files* command to list what is on the disk, delete anything unnecessary, and then give the Save command again.

... NONEXISTENT

A command or file was called for which does not exist. Check the spelling of the command or file name.

... EXISTS

An attempt was made to use a file name that already exists. Pick another name or delete the existing version of the file and try again.

WRONG NUMBER

A value which was supposed to be a number is not a number or the number specified is out of the allowable range.

TOO BIG

TOO MANY

TOO COMPLEX

TOO MUCH

The size of some file, problem, section, or whatever, is too great. Make smaller.

Mathematical Errors

A number printed as asterisks means the model number provided on the example line was not big enough for the value found in the table.

... MINOR ERRORS

This is a count of instances of too-large numbers and a variety of other mathematical errors including division by zero, taking the square root of a negative number, calculating a growth rate from zero, etc. If a calculation is in error, zero is produced as the result.

proof

The Proof command checks the working area for any words which are not in the T/Maker Dictionary or other specified dictionaries.

WHAT NEXT? **proof [with dictionary-1 with dictionary-2 etc.]**

The full command should be specified initially since prompts will not be made for optional dictionaries. **Proof with**, however, will result in a prompt for the dictionary name.

Dictionaries

Dictionaries are special compressed files prepared by the Amend command. They may not be edited.

Proof will always use the standard T/Maker dictionary (i.e., **T MAKER.DIC**), if present. For additional dictionaries, **with** should precede each name. The file-type **.DIC** will be appended to dictionary names specified without a file-type.

All dictionaries are expected to reside on the same drive as the command file **PROOF.TMK**. For optional dictionaries, a drive specification may be given.

proof with b:acronyms

Word Lists

A simple list of words may be used in place of a dictionary. However, the list must be in alphabetical order. To prepare such a list, all characters should be typed in the same case (all upper or all lower case). Then, the Sort command can be used to put the list in proper alphabetical order.

Definition of a Word

A word is a series of alphabetic characters ("a" through "z") that may include an apostrophe within it. However, if a word ends with an apostrophe followed by an "s," both the apostrophe and the "s" are not considered part of the word.

Capitalization

No distinction is made between upper and lower case when searching for words in the dictionary.

The Results of Proof

Proof prepares a "correction list" of words which were not found in the dictionaries used. Each line in this list begins with the characters **=!** followed by two occurrences of the word.

The first occurrence may be thought of as the current spelling of the word in the file. The second occurrence may be thought of as the spelling to be corrected and later substituted.

Proof also marks unidentified words in the text by preceding them with the characters **?=**.

Example

Working Area

Itt is now tme for all
good men to come to the aid
of their prty.

Result of: proof

=! Itt Itt
=? prty prty
=? tme tme

?=Itt is now ?=tme for all
good men to come to the aid
of their ?=prty.

correct

The Correct command corrects the working area based on the "correction list."

WHAT NEXT? **correct**

Preparing for Correct

After using the Proof command, the working area will contain a correction list of any unidentified words (see leftmost screen below).

You may react in one of four ways to each word in this list.

1. The word is an obvious misspelling.
(See case of "comnpy.")

You correct the spelling of the second occurrence of the word on the correction line.

The Correct command subsequently substitutes the second spelling of the word for the first spelling. The correction line is deleted from the file and any corresponding occurrences of ?= in the text are removed.

Note: If you are unsure of the correct spelling, the second occurrence of the word may be replaced by an ambiguous word reference. (See Spell command.) The Spell command may then be used to suggest alternatives.

2. The word is correct but should not be added to the dictionary. (See case of "Isuzu".)

You leave the correction line unchanged or delete it entirely.

The Correct command deletes the correction line (if you did not) and removes corresponding occurrences of ?= in the text.

3. The word is correct and should be added to the dictionary. (See case of "quadraphonic.")

You change the first position of the correction line from = to +.

The Correct command will remove corresponding occurrences of ?= from the text; however, the correction line will be left in the file. A subsequent use of the Amend command will add the word to the dictionary and remove the correction line.

4. The correctness of the word is impossible to determine without looking at the word in context. (See case of "rd.")

You can search for "?=" in the file to find the context and correct the problem with the Editor. To correct words which fell into the above cases before attacking those in this case, the second occurrence of the word in the correction list should be erased. This ensures that **Correct** will take no action for this word.

correct

Capitalization

Correct attempts to preserve capitalization in cases where one spelling is substituted for another. **Correct** differentiates four types of capitalization:

1. all in capitals
2. all in lower case
3. all in lower case except the first letter
4. none of the above

For the first three cases, **Correct** will mimic the capitalization of the word in the text when replacing it with another spelling. For the fourth case, **Correct** will use the exact spelling, including capitalization, as it is given on the correction line. Thus, when substituting for words with unusual capitalization, you should specify the exact spelling for case four.

= ! MicroMan MicroPerson

If only a single form of capitalization is to be used as the new spelling, then the second word in the correction line should be enclosed in single or double quotes.

= ! company "IBM"

If substitutions are to be made for only a single form of capitalization, then the first word in the correction line should be enclosed in single or double quotes. The second word in the correction line will be substituted exactly as it is entered (including capitalization).

= ! "JONES" MCBRIDE

= ! "Jones" McBride

Substituting Phrases

A phrase may be substituted for a word by placing the phrase appropriately on the correction line.

= ! us the company

Note: **Correct** may be used to replace words in a text file independently of other spelling commands by manually preparing the correction lines.

Note: **Correct** will not substitute spellings when a word on a correction line appears to be an ambiguous word reference. (See Spell command.)

Example

Result of: proof

```
=! compny      compny  
=! Isuzu      Isuzu  
=! quadraphonic  quadraphonic  
=! rd          rd
```

Our ?=compny is considering the purchase of an ?=Isuzu with ?=quadraphonic sound in the 3?=rd quarter.

Result of: editing

```
=! compny      company  
=! Isuzu      Isuzu  
+! quadraphonic  quadraphonic  
=! rd          rd
```

Our ?=compny is considering the purchase of an ?=Isuzu with ?=quadraphonic sound in the 3?=rd quarter.

Result of: correct

```
+! quadraphonic  quadraphonic  
=! rd
```

Our company is considering the purchase of an Isuzu with quadraphonic sound in the 3?=rd quarter.

amend

The Amend command creates or updates a dictionary from a list of corrections in the working area and other dictionaries.

WHAT NEXT? **amend [from dictionary-name + or - dictionary-name]
[to dictionary-name]**

All options should be specified in advance.

The Correction List

Correction lines in the working area beginning with "+!" denote words to be added to the dictionary.

Correction lines in the working area beginning with "-!" denote words to be deleted from the dictionary.

Lines of the above form will be removed from the working area after the dictionary has been amended.

Forms of Amend

Amend may be used in any of the following forms to achieve the described result. Dictionary names may be preceded by a drive designation (e.g., **b:master**). They are otherwise expected to reside on the same drive as the file **AMEND.TMK**.

amend
amends the dictionary TMAKER.DIC based on the correction list.

amend from temp
amends the dictionary TEMP.DIC based on the correction list.

amend to temp
creates the dictionary TEMP.DIC based on the correction list.

amend from y to x
creates the dictionary Y.DIC based on the correction list and X.DIC.

amend from x + y to z
creates the dictionary Z.DIC based on the correction list (if any) and the combination of the dictionaries X.DIC and Y.DIC.

amend from x - y to z

creates the dictionary Z.DIC by applying the correction list (if any) to the words in X.DIC which are not in Y.DIC.

When a dictionary is modified or created, the previous version is kept as a file with the file-type of ".BAK".

Saving Time

Amending a lengthy dictionary may take a number of minutes. The following procedure will save a considerable amount of time.

For the first amendments of the day use the command: **amend to temp**

For later amendments use the command: **amend from temp**

At the end of the day, use the command: **amend from temp + tmaker to tmaker**

During the day use: **proof with temp**

spell

The Spell command lists words from a dictionary on the screen or places them on correction lines.

WHAT NEXT? **spell [it or ambiguous-word-reference]**
[with dictionary-name]

An optional dictionary may be used; however, only the specified dictionary will be examined. The default dictionary is TMAKER.DIC.

Ambiguous Word References

An ambiguous word reference is composed of letters, question marks and asterisks. A question mark is used to represent one unknown letter. An asterisk is used to represent zero or more unknown letters. Thus, the ambiguous reference below would refer to all words that start with "b", have any second letter, have "m" as the third letter, and end with "ing".

b?m*ing

Ambiguous word references should be given as completely as possible to limit the range of the dictionary that needs to be searched.

When the Spell command is followed by an ambiguous word reference, words corresponding to the reference are listed on the screen.

The *it* Option

When *it* is specified instead of a word reference, the Spell command searches the working area for correction lines beginning with "=!" that also contain ambiguous word references.

For these correction lines, corresponding words in the dictionary are put on the line after the ambiguous reference until the line is full.

You can construct such correction lines from scratch or alter the second word of a correction line generated by **Proof** to look up words in a dictionary.

Example

Working Area

=! mi*pell
=! compatable compat*

Result of: **spell it**

=! mi*pell misspell
=! compatable compat* compatibility compatible

records

T/Maker's Data-Base commands deal with files of records. Such files contain a "record definition" which describes the manner in which a record is organized.

A record is a group of contiguous lines. On each line data values may be stored in fixed locations called "fields."

Record Definition

Any file of records must contain a record definition prior to the first record. The record definition shows:

- the identifying label for each line in the record
- the names and locations of fields in the record
- the manner in which the data in the field is justified

Record definitions are preceded by a line containing **< record >** and ended by a line containing **< end >**.

Record definitions look like masks used by Load and Unload, however, the mask is assumed to apply repetitively to each successive record in the file.

Example

```
< RECORD >
A { name } { age }
B { city }
< END >
```

Line Labels

Normally each line in a record begins with a unique label. The label extends from the first position of the line to the first brace. A close-brace may be used to delineate the label without beginning a field.

```
< RECORD >
+ } { category } { amount }
< END >
```

In some instances it is useful to have the same label for a number of lines in the record.

```
< RECORD >
1 { head }
2 { dependent1 }
2 { dependent2 }
2 { dependent3 }
< END >
```

This should only be done in situations like the one above where it remains nonetheless possible to identify the start of the new record because the first line does begin with a unique label.

Field Justification

Special symbols within the braces are used to denote the manner in which the value for the field should be justified.

records

Symbol	Interpretation
-	value centered in field
>	value right-justified with right-most limit of field
<	value left-justified with left-most limit of field
!	value left-justified and unused space after data eliminated ("squeezed")

The last symbol may destroy the validity of the record definition by shifting the positions occupied by following fields on the line. It should only be used to create temporary files intended for printing.

If a field contains no justification symbol, the same justification as the previous field is used.

For most record processing commands, right-justified values are assumed to be numeric. Fields justified in another fashion are assumed to be alphabetic. One should have a valid reason for violating this convention.

Optional Lines

Lines in a record may be made optional (i.e., they will not be generated if all fields are blank) by ending the line with a pair of close-braces.

```
<RECORD>
1 { head } }
2 { dependent1 } }
2 { dependent2 } }
2 { dependent3 } }
<END >
```

The first line in a record should never be made optional unless it is the only line in the record.

Non-Record Lines

Files of records may contain lines which are not parts of a record (e.g., headings). By adding a record definition to an existing list or table, data base commands may be used on the list or table.

Multiple Record Definitions

A file may contain more than one record definition. In such cases, the first record definition encountered in the working area (not before it) or the first record definition in a disk file will be used. The Update command, however, always uses the first definition in the file.

Field Names

The names of fields are placed within the braces that define the location of the field. Names should not contain any of the following special symbols:

- < > = ! # + - * / !

In general, it is best to begin names with a letter (alphabet character) and to pick names which are not themselves data values or keywords used with other commands such as "when" or "contains" (used by Select). Consider that short names conserve space.

If a set of braces contains no name, its name is assumed to be "one higher than the previous name" (see Load - "Data Files").

If a name is too long to fit within the corresponding braces, the conventions described in "Long Names in Masks" may be used (see Unload). For fields which are only one character long, double number signs are used immediately after the open brace.

```
{ ##gender }
```

Note: Data Transfer commands do not acknowledge this convention.

Each field should have a unique name.

Note: This rule is sometimes profitably violated when creating temporary files with the Select command.

update

The Update command permits files of records to be altered.

WHAT NEXT? *update*

General

The working file to be updated must contain a record definition and, optionally, form definitions. When no form definition is present, the record definition itself is used as a form.

Form Definition

A form is a screen image used for the purpose of entering, updating, or displaying a record (see Example below).

A form definition begins with a line containing **<form>** and ends with a line containing **<end>**.

Form definitions are placed in the working area ahead of the correct record definition.

A form must fit entirely on one screen. Up to ten different forms may be used.

Forms may consist of lines of text or decoration and fields. Fields are enclosed in braces and contain the name of the field.

The space allotted for a field in a form should be larger than or equal to the space allotted in the record. Data can be

entered up to the smaller of the two allotments.

Ideally, the first line of a form is left blank to permit space for generated messages.

New Files

When creating a new file, records are normally placed immediately after the record definition. A line containing **<here>** can be put in the file to mark where the first record should be placed.

Records can only be added next to other records. If records are to be entered in different groups separated by non-record lines, dummy records must be put in using the Editor. These serve as seeds from which the groups can grow.

The CHANGED Field

If one of the fields in the record is named **changed** or **CHANGED**, it will be set to the value **1** for any record which is changed via the Update command.

Example

Working Area

```
< FORM >
Name: {name}
Room: {room}
< END >
< RECORD >
1 {name}      } { room }   { changed }
< END >
1 John Doe    X99
```

Initial Screen Produced by the Update Command

```
Name: John Doe
Room: X99
```

Revise Records

	MINE
Insert character	Ins
Delete character	Del
Erase line	F10
Delete spaces to next character	Esc Ins
Destructive backspace	←
Erase to end of field	Esc Del
Delete record	Esc F10
Insert record	Esc F9

Move a Record

Clear the buffer	Esc F8
Copy to the buffer	F7
Move to the buffer	F8
Insert the buffer	Esc F7

Search

Search for string	F5
Continue search	F6

Change Modes

Insert mode	Esc I
Change form	Esc F

Move the Cursor

	MINE
Up	↑
Down	↓
Left	←
Right	→
Home	Home
Next word	End
Next field	→
Previous field	Esc →
Next line	←

Move the Screen

Home	Esc Home
First record	Esc ↑
Previous record	PgUp
Next record/ First field	Esc ←
Next record	PgDn
Last record	Esc ↓

Others

Quit update	F1
Repeat last keystrokes	Ctrl K
Information	F2
Redraw the screen	Esc F2

Note: When a keystroke for the Editor is changed with T/Modify, the corresponding keystroke for the Update command will also be changed.

update

Non-PC Version

Revise Records

MINE

Insert character	CTRL	W	_____
Delete character	CTRL	E	_____
Erase line	CTRL	T	_____
Delete spaces to next character ..	CTRL	U	_____
Destructive backspace	DEL		_____
Erase to end of field ..	CTRL	Y	_____
Delete record	ESC	<	_____
Insert record	ESC	>	_____

Move a Record

Clear the buffer	CTRL	C	_____
Copy to the buffer ...	CTRL	F	_____
Move to the buffer ...	CTRL	G	_____
Insert the buffer	CTRL	D	_____

Search

Search for string	ESC	,	_____
Continue search	ESC	&	_____

Change Modes

Insert mode	ESC	I	_____
Change form	ESC	F	_____

Note: When a keystroke for the Editor is changed with T/Modify, the corresponding keystroke for the Update command will also be changed.

Move the Cursor

MINE

Up	CTRL	K	_____
Down	CTRL	J	_____
Left	CTRL	H	_____
Right	CTRL	L	_____
Home	CTRL	O	_____
Next word	CTRL	P	_____
Next field		TAB	_____
Previous field	ESC	TAB	_____
Next line		RET	_____

Move the Screen

Home	ESC	CTRL	O	_____
First record	ESC	A		_____
Previous record	ESC	CTRL	K	_____
Next record/ First field	ESC		RET	_____
Next record	ESC	CTRL	J	_____
Last record	ESC		Z	_____

Others

Quit the update	CTRL	Q	_____
Repeat last keystrokes	CTRL	A	_____
Information	ESC	?	_____
Redraw the screen	ESC	#	_____
See Addenda for IBM Users keystrokes			

order

The Order command organizes the records in the working area alphabetically or numerically in either ascending or descending order. Records are ordered on the basis of a field in the record.

WHAT NEXT? **order [options, if any] field-name**

General

Any lines in the file which are not a part of records form barriers. Records are never moved across such barriers (see Example below).

Options

Select one or none from each of the two following categories.

ascending ascending order
descending descending order

characters orders characters

alphabetically

numbers orders numerically

Options may be abbreviated to one or more letters provided the given spelling is not also the name of the field (e.g., if a field is named "d", "descending" can be abbreviated to "de").

Defaults

If **ascending** or **descending** is not specified, **ascending** is assumed.

If **numbers** or **characters** is not specified, **numbers** is assumed if the field used as the basis of the ordering is right-justified. Otherwise, **characters** is assumed.

Note: To order on multiple fields, order first on the least important field, then the next most important, etc.

Example

Working Area (records are unshaded)

< RECORD >		
+ { name < } { age > }		
< END >		
Males		
+ Bob	27	
+ Tom	32	
Females		
+ Alice	23	
+ Susan	27	
+ Mary	44	

Result of: **order d age**

< RECORD >		
+ { name < } { age > }		
< END >		
Males		
+ Tom	32	
+ Bob	27	
Females		
+ Mary	44	
+ Susan	27	
+ Alice	23	

set

The Set command sets fields in a record to values which may be alphabetic or calculated from other fields in the record.

WHAT NEXT? **set**

Rules Definition

The working area must contain a definition of the rules to be applied.

The definition begins with a line containing **<rules>** and ends with a line containing **<end>**. Rules are placed between the two lines. The definition is located ahead of the corresponding record definition.

A file may contain multiple sets of rules, however, the Set command will impose only the first set of rules encountered.

General Form of a Rule

A rule is written on one line and has the following general form.

field-name [model-number [zero-value]] =
expression [**when** condition] **end**

The model number is optional and is formed as for the Compute command to control the accuracy and presentation of a resulting figure.

The model number may optionally be followed by a character string used to represent exact zeros.

The condition is also optional. If no condition is present, **when** should not be placed on the line.

If the rule is followed by a comment, **end** must be placed prior to the start of the comment.

For each record in the file for which the condition is true, the field name specified is set equal to the expression.

A number of rules can be made conditional on the basis of a single condition by the following form.

when condition
rule1
rule2
etc.

Each rule in the above may also have its own specific condition. The condition for the group of rules continues to apply until a similar form is encountered.

To escape from this form, specify a condition which is always true.

when 1 < 2

Expressions and Conditions

Expressions and conditions are formed using normal algebra and logic with the following elements:

a. Field Names

Must be spelled exactly as in the record without any imbedded spaces.

b. Constants

Must be enclosed in single or double quotes if the constant contains an embedded space, a single or double quote, a special character, or if the constant is the spelling of a field name or operator.

c. Operators

Assuming no parentheses, operators are evaluated in the precedence order shown below. Operators in the same group have equal precedence.

!	concatenation
*	multiplication
/	division
+	addition
-	subtraction
<	less than
<=	less than or equal to
=	equal to
>>	not equal to
>=	greater than or equal to
>	greater than

set

contains	e.g., "the" contains "h"
within	e.g., "h" within "the"
among	e.g., "Single" among "Married Single"
not	logical not
and	logical and
or	logical or

d. Parentheses

May be used to override normal precedence and should be used in any confusing case.

Evaluation of:

< <= = <> >= >

When any of the above operators are used, the evaluation may be made as a character comparison or a numeric comparison. A numeric comparison is made only if both fields are right-justified or numeric constants (i.e., not enclosed in quotes and beginning like a number).

Otherwise a character comparison is made.

This convention can be overridden by preceding the operator with the keyword **numeric** or **char** to denote the desired type of test.

Note that the following conditions are both true.

0 char < > 0.0 0 numeric = 0.0

Truth and Falsehood

A value by itself is true if it converts to a non-zero number, otherwise it is false (e.g., "1 and 3" is true).

A logical comparison yields a value of "1" if it is true otherwise "0".

Relative Record Number

The word **number** may be used in an expression in place of a field or constant to refer to the relative number of the record in the file (1, 2, 3, etc.).

id = **number**

Examples

total 99,999.99 " " = **quantity** * **price** **end**

Mathematical assignment using a model number and a zero value of spaces.

category = **A** when **age** < = 20 and **tickets** > 5 **end**

Character assignment with a condition.

state = "New York" when **zip numeric** > = 10001 and **zip numeric** < 10200 **end**

Character assignment with a condition. Presumably the field zip is shown as left-justified thus "numeric" is needed to ensure the correct comparison.

positives = (**answer1** = Y) + (**answer2** = Y) + (**answer3** = Y) **end**

A condition returns "1" if true, thus "positives" will be set to the number of answers which are "Y".

status = "ok" when **type** among "dealer distributor oem" **end**

"Among" tests for one of a series of possibilities separated by spaces.

price = **cost** + 1.67 when **category** not < 3 **end**

Example use of "not" to negate a particular comparison.

control = ERROR when not (**rate** > = 2.50 and **rate** < = 35.00) **end**

Example use of "not" to negate a condition.

group = 1 when **name** > E and **name** < K **end**

Sets "group" to 1 for names beginning with "F" through "J".

condition = RED when **expenses** > **income** - 1000 **end**

Mathematical operators may also be used in the formation of conditions.

select

The Select command selects particular records and fields from an existing file of records.

WHAT NEXT? *select disk-file-name or it [options, if any] end*

Selecting from a Disk File

If the working area has no record definition, the definition from the disk file is inserted.

The working area may contain a record definition different from the disk file. Records selected from the disk file will be reorganized according to the new definition.

Selected records are placed:

- after a line containing **<here>**;
- after the record definition.

Selecting IT

When *it* is specified, records are selected from those in the working area itself. Lines which are not parts of records are left in place.

When the *it* option is used, a **<continue>** specification in the working area is ignored.

Options

list	Lists the names of the fields in the record.
from 5	Selects records only from the 5th record forward.
to 25	Selects records up to and including the 25th record.
when condition	Selects records only if the logical condition is true. If used, this option must be the last one. (See Set command for details of logical conditions.)

<CONTINUE> file-name

When selecting from a disk file, the last line of the disk file may contain **<continue>** followed by a file name. The named file will be accepted as a continuation of the original. This file may itself be continued.

The Record Definitions of all such linked files must be identical.

Example

Disk File "PERSONS"

```
<RECORD>
1 { name      }
2 { age>} { phone   }
<END>
1 John Doe
2    47 335-6521
1 Mary Ellen
2    43 532-9911
1 Robert Smith
2    55 777-8080
```

Working Area

<RECORD>		{ name } { phone }	<END>
Name	< HERE >	Phone number	
< HERE >			
Name	< HERE >	Phone number	
John Doe		335-6521	
Robert Smith		777-8080	

Result of: **select persons
when age > 45 end**

complete

The Complete command completes records in the working area with information from records in a disk file. A specified field is used to determine which records on disk match those in the working area.

WHAT NEXT? *complete disk-file-name matching-field-name*

Matching Records

For a record in the working area to match a record on disk:

- the record in the working area must have some non-blank value for the matching field.
- the value for the field in the working area must be identical with the value in the disk record. (**Note:** 05 is not identical with 5)

One record on disk may match none, one, or more than one record in the working area.

Completing Fields

When a match is made, any field in the working area record, which is also in the disk record, will take the value of the disk record.

Fields defined in the working area record, which are not defined in the disk record, will keep their initial values.

If no match is made for a record, the record will be unchanged.

<CONTINUE> file-name

The last line in the disk file may contain <**continue**> followed by a file name. The named file will be accepted as a continuation of the original. This file itself may be continued.

Example

Disk File "TITLES"

```
<RECORD>
+ {project} {title}
<END>
+ ROAD      Road Constr.
+ HOUSE     House Constr.
+ BRIDGE    Bridge Constr.
+ PAVE      Street Paving
<CONTINUE> titles.2
```

Working Area

```
<RECORD>
1{name} {project} {title}
<END>
1 Al      PAVE
1 Mary    HOUSE
1 Peter   STAND
1 Tom     PAVE
```

Result of:
complete titles project

```
<RECORD>
1{name} {project} {title}
<END>
1 Al      PAVE      Street Paving
1 Mary    HOUSE     House Constr.
1 Peter   STAND
1 Tom     PAVE      Street Paving
```

group

The Group command groups together records in the working area. A group is formed of records which have identical values for the fields specified.

WHAT NEXT? **group [list] [field-name 1, field-name 2, etc.] end**

General

The command **group end** would group all records into one record.

Lines which are not a part of records are left in place.

The List Option

The List option lists the names of the fields in the record.

The Grouped Records

For each group, one record results.

For any fields which are left-justified or centered, the resulting group record will contain the values of the first record encountered for the group.

For any fields which are right-justified (and not used as a basis for defining groups), the resulting group record will contain the sum of the values for all records in the group.

The COUNT Field

If a field in the record has the name **count** or **COUNT**, it will be set to one for any record which has a blank value for the field.

When records are grouped, the **COUNT** field will be summed like any other right-justified field. Thus, this field can be used to count the number of records in each group.

Example

Working Area

```
<RECORD>
+ {name } {dept } {age >} {count }
END
+ Bob      PERS      27
+ Tom      FIN       32
+ Alice    PERS      23
+ Susan    FIN       27
+ Mary     FIN       44
```

Result of: **group dept end**

```
<RECORD>
+ {name } {dept } {age >} {count }
END
+ Bob      PERS      50      2
+ Tom      FIN       103     3
```

error messages

(Data Base Only)

General Errors

Most error messages generated by the Data-Base commands are self-explanatory. Care should be taken to spell field names correctly, since an incorrectly spelled name may be treated as a constant and not diagnosed. A few non-trivial problems are described below.

(ALMOST) OUT OF SPACE

Data-Base commands will complain about running out of space if there is insufficient space for one more complete record (in addition to the minimum required for other commands).

PROBLEM IS TOO COMPLEX

The combination of the definitions in use by the command (i.e. record, form, and rules) are too complex. To reduce the complexity of the problem:

- keep identifying labels and field names short
- reduce the length of fields to a minimum
- eliminate any unused fields entirely from the record

If the Set command is the problem, consider breaking the rules into two groups and using Set once for each group.

INVALID EXPRESSION*

An expression or condition is improperly formed. Check the spellings of field names and operators as well as their placement within the expression.

UNMATCHED PARENTHESES*

Left and right parentheses do not match in an expression or condition.

TOO MANY EX- OR IMPLICIT PARENTHESES*

An expression or condition generates too many nested parentheses. For example:

$A + B + C + D + E + F$

is implicitly equivalent to

$((((A + B) + C) + D) + E) + F$

If there were 20 pluses instead of just 5, this would be a problem. An explicit and more balanced use of parentheses would be:

$((A + B) + (C + D)) + (E + F)$

... VALUES TRUNCATED

Some values were larger than the fields intended to hold them. Such values are shortened on the right in order to fit.

NEED FEWER RECORDS OR SHORTER FIELDS

There are too many unique groups for the Group command. Grouping on fewer or shorter fields or grouping a fewer number of records may solve the problem.

INADEQUATE SPACE FOR SUMMED VALUES

While using the Group command, the summations of some field produced numbers which did not fit in the space provided. The problematic fields will be filled with asterisks.

RECORDS TOO BIG TO ORDER

The Order command may not work with unusually large records.

MINOR ERRORS

The Set command may produce the same types of minor mathematical errors as the Compute command.

*When using the Set command, the top line on the screen is the offending rule.

compute

The *Compute* command carries out mathematical calculations on the working area.

WHAT NEXT? *compute [c c c... as needed-used for carrying values from the bottom of a table back to the top]*

All mathematical calculations are first laid out on the screen in table form (while in the editor) and then calculated by invoking the *Compute* command. This table must include an "example line" and numbers must be properly located beneath it. An optional "zero values" line may be specified. The first seven positions of each line are reserved for special purposes and may not be used for headings or numbers.

Example Line

The example line locates the zones in a table that will be used for columns of numbers and provides models of how those numbers should be presented ("model numbers").

- *ex* in positions 1 and 2 followed by model numbers after position 7
ex 9,999 .99.9 ,99
- May contain up to 25 model numbers
- Can be installed at any point in table in order to change features of presentation

Model Numbers

- Separated from each other by one or more spaces
- Consist of strings of nines (or other characters), optional commas, optional decimal point
- May contain up to 13 nines
- Will not print numbers below it when started by a comma (third column of example line above)

Placement of Numbers in Table

- Only within left and right limits set by model numbers in example line
- Must be contiguous (no blank spaces) including minus signs.
- Will be automatically aligned when entered correctly.

	Right	Wrong
<i>ex</i>	999,999.99	999,999
<i>+ 12.3</i>	- 45	5678
<i>+ 12,678.9</i>	1,234	
<i>+ -2156</i>		

Zero Values Line

Though the example line specifies how numbers should be presented, it is often desirable that null values are given special treatment. The Zero Values Line provides this option.

- Follows example line
- *zv* in positions 1 and 2 followed by symbols designating how null values are to be presented

*ex 999.99 999.99 9.9999
zv - none*

Note: Values falling close to, but not exactly upon, zero are always presented in the fashion prescribed by the model nines, even when all of the digits shown are zeros.

compute

General Calculations

General calculations may be laid out horizontally (across columns), vertically (down rows), or across tables (by employing the *Combine* command). They consist of mathematical operators followed by a single terminator symbol placed in relation to a row or column.

- **mathematical operators**—define the operation to be performed upon the values to which they refer
 - addition (+) multiplication (*)
 - subtraction (–) division (/)
- **terminator symbols**—call for a result to be produced and printed. An equal sign (=) produces the following calculating sequence:
 - (1) *Compute* begins with zero,

- (2) sums all numbers associated with plus signs (+), if any;
- (3) subtracts all numbers with minus signs (–), if any;
- (4) multiplies by all numbers associated with multiplication signs (*), if any;
- (5) and, finally, divides by all numbers associated with division signs (/), if any.

As an equation, *Compute* would be:

$$[(\text{Sum of } +\text{'s}) - (\text{Sum of } -\text{'s})] \times (\text{Product of } *\text{'s}) \\ / (\text{Product of } /\text{'s})$$

Note: If a general calculation contains no plus or minus signs, the result must be zero.

Terminator Symbols

A general calculation may be terminated by an equal sign (which generates the simple result) or by one of the other symbols listed below. Each of these additional terminator symbols can be thought of as performing one last calculation upon the simple result before generating an output figure.

Symbol	Modification
=	simple result
%	result multiplied by 100
#	result divided by 100
sfh	result subtracted from 100
sfo	result subtracted from 1
rec	reciprocal of result
sqr	square root of result
abs	absolute value of result
exp	e to power of result
log	natural log of result
l10	log to base 10 of result
sin	sine of result (radians assumed)
cos	cosine of result (radians assumed)
tan	tangent of result (radians assumed)
atn	arctangent of result (radians produced)

sgn	sign of result (+1, 0, –1)
ply	maximum of zero and result (positives only)
nly	maximum of negative of result and zero (negatives only)
avr	result divided by number of plus signs in equation (if equation only has plus signs, result would be the average)

Special Terminators

With special terminator symbols, + and *, are used to flag or identify values. The result is reported at the spot indicated by the placement of the special terminator symbol.

+ * max	selects maximum of two numbers identified by the + and *
+ * min	selects minimum of two numbers identified by the + and *
+ * ~	produces percent change in going from value identified by + to value identified by *
+ * ^	produces value identified by + symbol raised to power of the value identified by *
+ + cnt	produces count of plus signs in equation

Note: For three-letter terminators, the position of the first letter determines the appropriate strip or column.

compute

Vertical Calculation Strips

Individual calculation equations laid out in a single position are called Strips

- may only employ General Calculations
- must be located in positions 1-7.
- consist of one or more mathematical operators placed above terminator symbol
- can be linked together
- cause the calculation to be carried out on all columns of the table

Once a terminator symbol has been used in a Strip, a new calculation may be started below it in the same Strip. If more than one Calculation Strip terminates on the same row, the output result will be the sum of each Strip's result.

ex		999.99
+	Item A	40.00
+	Item B	60.00
= +	Total Price	100.00
*	Percent Tax	5.00
= #	Price With Tax	105.00

(Figures in bold face are calculated)

Calculation Lines

Individual calculations laid out on single lines are called Horizontal Calculation Lines.

- may appear anywhere in a table
- may apply to the rows below
- always begin with a Control Code in positions 1 and 2 which specifies the set of rows upon which this calculation should be carried out.
- always have a Sequence Number next (0 to 20) which specifies the order in which this calculation will be carried out relative to other Calcula-

tion Lines. If a Calculation Line reuses a previously used Sequence Number, the old Calculation Line is replaced by the new.

- symbols must be placed in zones defined by the example line (see shaded areas)

When using General Calculations on a Calculation Line, the Terminator Symbol must be placed to the right of the other symbols. After a General Calculation has been terminated, a new one may be started on the same line. The order of such calculations is left to right.

	List Price	Price Paid	Percent Discount
ex	999	9999	999
ac1	/	+	% + sfh
+	50	40	20

Calculation Line Control Codes

- ac** Always Compute—applies to all subsequent rows
jc Just Compute—applies only to the next row
uc Usually Compute—applies to all subsequent rows except those derived from other rows
rc Rarely Compute—applies only to subsequent rows derived from other rows
cc Cancel Compute—all previous Calculation Lines no longer apply below. To cancel the application of a single Calculation Line, follow cc with the Sequence Number of the Calculation Line to be cancelled
sc Suspend Compute—all previous Calculation Lines will not apply to next row. To suspend a single Calculation Line, follow the code with the Sequence Number of the Calculation Line to be suspended

compute

Restricted Calculations

Restricted Calculations may be used only in computations laid out horizontally, i.e., only upon Horizontal Calculation Lines.

Placing Constants into a Calculation

Constant values can be placed into a calculation by employing a special type of Calculation Line. This line consists of a calculation Control Code followed by a Sequence Number followed by a Special Symbol designating what is to be done with the constants entered into this Calculation Line. Different constants can be entered into the same Calculation Line. Columns left blank or presenting a zero value will be left unaffected by the constant computation.

ex	999	9999	9999	9999
uc1*		5	6	0

For example, in the case above, this Calculation Line would cause the value in the second column of the row to be multiplied by five and the value in the third column of the row to be multiplied by six. The values in the first and fourth columns of the row would be unaffected.

Special Symbols for Constants

- = column set equal to constant
- + constant added to value for column
- constant subtracted from value for column
- * value for column multiplied by constant
- / value for column divided by constant
- % value for column divided by constant and result multiplied by 100

#	value for column multiplied by constant and result divided by 100
max	maximum of value for column and constant becomes new value for column
min	minimum of value for column and constant becomes new value for column.
^	value for column raised to power of constant.

Store, Fetch, Pass, Round

The symbols described here may be given on a Calculation Line by themselves or they may be mixed with the symbols of General Calculations.

st?	The interim value for the column is stored for future use. A single letter, in place of the question mark, becomes the stored value's "name."
ft?	The stored value (a one-letter name in place of the question mark) is fetched and placed into the column associated with the site of ft?
pas	The interim value for the column is "passed" to the next row. The value passed thus becomes the initial value for the same column in the next row. pas values will override other values for the initial value, e.g., values derived from a Vertical Calculation Strip.

rnd	The interim value for the column is rounded to the accuracy of the model number that applies to it and calculations using this value will employ the rounded instead of the unrounded value.
-----	--

Restricted calculations continued on next page

compute

Special Notations

A number of Special Notations may be used on Calculation Lines. These Special Notations must appear on a line by themselves. The symbols need not appear in the order shown below.

+ + + + mvr

The value of a + column is moved to the + column to its right.

+ + + + mvl

The value of a + column is moved to the + column to its left.

+ + + + cum

+ columns are converted to a cumulative distribution (left to right).

+ + + + chg

+ columns are converted to represent the difference (or change) between a + column and the column to its left.

+ + + + pch

+ columns are converted to represent the percent change between a + column and the + column to its left.

+ + + + pct

+ columns are converted to percents of the *pct* column.

+ + + + prj

The leftmost nonzero + column is projected to the right based on the growth rate in the *prj* column.

+ + + + inc

The leftmost nonzero + column is projected to the right based on the additions of the increment in the *inc* column.

- * * + grw

The growth rate in going from the - column to the + column is put in the *grw* column. * is used to mark intervening periods.

- / + grw

The growth rate in going from the - column to the + column is put in the *grw* column. The / column should contain the number of periods over which the growth has occurred.

dis + + net

The discount rate in the *dis* column is applied to the + columns to derive the net present value which is put in the *net* column.

The Order of Calculations

Tables are calculated one row at a time proceeding from the top to the bottom of the table.

For each row:

- If the row is derived from other rows,

the appropriate Calculation Strip is used to derive all columns of the row.

- The appropriate Calculation Lines as determined by Control Codes, are then applied in order as determined by Sequence Numbers.

an example

ex		9999	9999
ac 2	Calculation Line 2 (applies to all rows)		
uc 3	Calculation Line 3 (applies to data rows)		
rc 4	Calculation Line 4 (applies to derived rows)		
+	Row A (Data)	✓	✓
+	Row B (Data)	✓	✓
=	Row C (Derived)	✓	✓
jc 1	Calculation Line 1 (applies to next row)		
★	Row D (Data)	✓	✓
=	Row E (Derived)	✓	✓

compute

A Complete Example

	BUDGET FORECAST				Year Total
	January	February	March	April through December	
ex	999,999	999,999	999,999	999,999	999,999
zv		-	-	-	-
uc1				sta	fta
uc2*					9
uc3	+	+	+	+	=
+	Mortgage Payment	456	456	456	4,104
+	Car Payment	340	340	-	680
+	Other	600	625	650	5,850
= - +	Total Expenses	1,396	1,421	1,106	13,877
+	Regular Salary	1,700	1,850	1,850	16,650
+	Consulting	350	400	450	4,050
= + /	Total Income	2,050	2,250	2,300	27,300
	Expenses as Percent of Income	68	63	48	48
+ =	Net Income	654	829	1,194	10,746
jc4	cum +	+	+	+	
=	Cumulative	654	1,483	2,677	13,423
					13,423

(Figures in bold are calculated.)

Explanation for the Order of Calculations

- Row A is processed by applying Calculation Lines 2 and 3.
- Row B is processed by applying Calculation Lines 2 and 3.
- a. Row C is derived from rows A and B according to the Calculation Strip.
- b. Row C is processed by applying Calculation Lines 2 and 4.
- Row D is processed by applying Calculation Lines 1, 2, and 3.
- a. Row E is derived from rows C and D according to the Calculation Strip.
- b. Row E is processed by applying Calculation Lines 2 and 4.

Explanation for a Complete Example

The first three Calculation Lines apply only to data rows because their Control Code is given as *usually compute*. The first Calculation Line sets the fourth

column equal to the third (see Store and Fetch). The second Calculation Line multiplies the value on the fourth column by nine thus making the estimate for the fourth column equal to nine times the March figure (see Placing Constants). The third Calculation Line sums the first four columns into the fifth.

Various Calculation Strips are used to derive rows of interest. The first three Calculation Lines apply to none of these derived rows. Just ahead of the last row, however, a Calculation Line has been added which will apply to the last row. It converts column values to their cumulatives (see Special Notations). Thus, after the Calculation Strip sets the Cumulative Row to the same figures as the Net Income row, Calculation Line 4 modifies the values in the first four columns to achieve the cumulative.

clean

The *Clean* command removes from a table in the working area the example line, the zero values line, the column names line (used by the Bar command), calculation lines and strips preparing the table for final printing. Print design commands are unaffected by the *Clean* command.

WHAT NEXT? *clean*

Notes

1. Design commands placed within a table tend to muddle its appearance. One tip is to disguise design commands as ordinary lines of text by beginning the commands in position eight instead of position one. After *Clean* removes the first seven positions, the formerly-disguised commands become legitimate design commands.
2. The *.clean* print design command provides another, and perhaps more powerful, approach to cleaning.
3. *Clean* never removes more than the first seven positions of a row even when a terminator symbol extends beyond position seven (e.g., *avr* with the *a* in position six). If such a situation is unavoidable, an easy solution is to start the text of the table in position 15 and use *Clean* twice in succession.

An Example of Clean

Before

Profit Analysis				
	Item	Cost	Sales	Profit
.end		999,999.99	999,999.99	999,999.99
ex		-	+	=
ac1				
+	Windows	125,167.80	242,234.50	117,066.70
+	Mirrors	66,145.10	110,499.10	44,354.00
+	Storm Doors	48,091.23	57,120.11	9,028.88
.block				
=	Total	239,404.13	409,853.71	170,449.58
.end				

After

Profit Analysis				
	Item	Cost	Sales	Profit
.end		125,167.80	242,234.50	117,066.70
	Mirrors	66,145.10	110,499.10	44,354.00
	Storm Doors	48,091.23	57,120.11	9,028.88
.block				
	Total	239,404.13	409,853.71	170,449.58
.end				

combine

The *Combine* command combines a table on disk with a table in the working area, generating a new table in the working area.

WHAT NEXT? **combine** *disk-file-name* *disk-file-symbols*
working-area-symbols

- Each parameter of the *Combine* command is separated by a space, but no spaces should appear within a parameter.
- The combined table's numbers are presented according to the model numbers provided by the working area's example line.

Specifying the Calculation

Following the *Combine* command, the user must specify three parameters.

1. The name of the disk file with which the working area table is to be combined.
(The second and third are General Calculation symbols telling how the two tables should be combined.)
2. The mathematical operator(s) which determines what is to be done with a cell value in the disk file table.
3. a. The mathematical operator(s) which determines what is to be done with the cell-value of the working-area table.
b. A terminator symbol specifying the result to be produced. This result becomes the cell-value of the newly generated table in the working area.

The General Calculation is applied to the cells of the disk file and the working area on a corresponding cell-by-cell basis.

Examples

To sum the numbers in the working area with those in a disk file named *Sales.mar*:

combine sales.mar + + =

To average the numbers in the working area with those in a disk file named *tab.2*:

combine tab.2 + +avr

To take the numbers in the working file as a percent of the numbers in a disk file named *actual*:

combine actual / +%

To make the table in memory equal to the sum of disk files named *jan*, *feb*, and *mar*:

**combine jan + =
combine feb + + =
combine mar + + =**

Noncongruent Tables

Ordinarily, *Combine* is used on tables having the same numbers of rows and columns. If, however, one table has fewer columns or fewer rows, values will be combined only where corresponding values exist. Where the calculation is not applied, the working area's values will remain unchanged.

bar

Bar prepares a bar chart based upon the values in either one column or one row of a table. This table must be the working file and must also conform to the format requirements of the *Compute* command. (See below for options.)

WHAT NEXT? **bar [column-or-row-name or *it*] [options] end**
The user can choose how the bar chart will be presented by experimenting with various options. Each variation can be examined by specifying one or another option and pressing the carriage return key. The *end* should not be typed until a return to the Universal Prompt is desired.

Bar charts may either be displayed or placed into the working file for editing and printing.

Designating the Row or Column to Be Charted

The row or column of values to be charted is designated by name. In most cases, it is sufficient to provide the name appearing at the top of a column or the name at the side of a row.

A column name will cause values in that column to be rendered into bars; a row name will cause values in that row to be rendered into bars. Names containing a blank space must be enclosed by quotation marks and must appear exactly as they do in the table. The name *it* may be used to chart the row of values at the top of the screen (or top of the frame).

Specifying Values to Be Charted

The options below control which values will be charted in a given row or column. When neither option is specified, the default is *some*.

some When a column of values is to be charted, this option will include only those rows with + in the first position.

When a row of values is to be charted, this option will exclude columns whose model numbers begin with a zero. All other columns will be included in the chart.

all All values in the column or row are charted.

Overriding the Row or Column Designation

Normally, the location of the name (at the top of a column or at the side of a row) will determine whether a row or a column of values is to be charted. This can be overridden by the following options:

row forces *Bar* to use a row of values

col forces *Bar* to use a column of values

Examples of the Values Charted by Various Commands

bar Supplies all	bar Increase
bar Total	bar 1979 all
Expenses	
ex	999,999
cn	1978
ac1	-
+	Consultants 42,521
+	Supplies 11,256
+	Rent 8,200
+	Other 17,235
Total	83,057
	100,599
	16,874

Column Names

A line beginning with *cn* may be included in a table to specify names to be used for the columns when they are presented in a bar chart. (See example above.)

bar

Presentation Options

Presentation Options control various features of the bar chart's appearance.

numbers causes numbers identifying bars to be placed in a column down the left side of the chart

nonumbers returns the bars to an unnumbered condition

names 11 provides 11 positions for the names of bars (up to 100 positions may be requested)

names 0 suppresses the names of the bars altogether

values 6 provides six positions for the values corresponding to bars (up to 100 may be requested)

values 0 suppresses showing the values

min -2250 sets the lower limit of the scale

len 50 sets the number of positions spanning the lower and upper limit

max 10000 sets the upper limit of the scale

nobars suppresses showing bars
bars bars restored

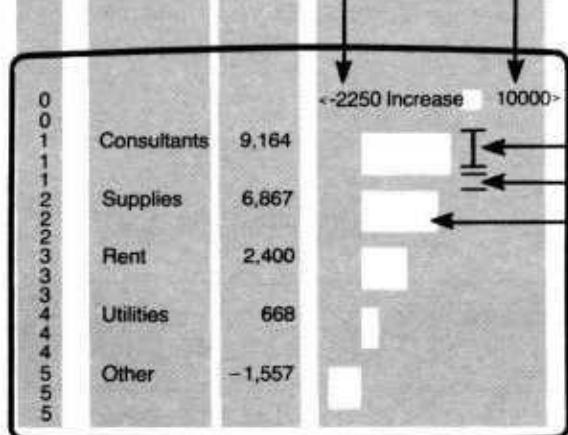
width 2 sets the width of bars to two lines

space 1 sets the spacing between bars to one line

char □ sets the character used to draw bars

short shows only one screen of the bar chart

long scrolls all bars onto the screen (the space bar may be used to interrupt the scrolling)



Disposition Options

Disposition Options control the bar chart's fate. Ordinarily these options should be used only when the *Bar* command has been invoked at the top of the working file.

replace replaces the working file with the current chart

insert inserts the current chart ahead of the first line in the working file

stack stacks the bars of the current chart onto the end of the bars of the chart previously inserted

restart deletes the charts ahead of the initial working file

cut deletes the original working file while keeping any inserted charts

Use the following options for reviewing the result of commands entered:

file displays the initial working file on the screen

chart displays the charts inserted ahead of the initial working file

arrange/match

The *Arrange* command reorders the sequence of columns for lines in the working area.

WHAT NEXT? **arrange** [left-position right-position, etc] **end**

The columns to be reordered are identified by pairs of numbers defining their present boundaries. The first number marks the left boundary and the second number marks the right boundary of a column to be moved.

The new order of columns is determined by the sequence of number-pairs

following the *Arrange* command: e.g., in the case shown below, the column defined by the number-pair 30-35 will come first occupying positions 1-6. Columns which are not described by a number-pair will be lost.

Note: Columns of blanks may be moved around to provide white space in a table. Design commands are unaffected by *Arrange*.

An Example of Arrange—**arrange 30 35 25 28 10 19 end**

BEFORE							AFTER								
1	5	10	15	20	25	30	35	1	5	10	15	20	25	30	35
Male	Single		17	Ray				Ray	17	Single					
Male	Married		42	Bob				Bob	42	Married					

The *Match* command contrasts the working-area with a file on disk.

WHAT NEXT? **match disk-file-name**

This command prepares a summary of the differences between the two files which then becomes the working area.

The summary shows the origin of unmatched lines as **W>** for a line in the working area not on the disk file or **D>** for a line in the disk file not present in the

working area. The line numbers of unmatched lines also are shown.

Blank lines are ignored throughout and no attempt is made to match them. The user is responsible for contrasting the appropriate lines.

An Example of Match

THE DISK FILE				THE WORKING AREA				THE NEW WORKING AREA (Summary)			
Table of Weights				Table of Weights				5 D> John 189 180 175 7 W> Alice 8 25 34			
	1980	1981	1982		1980	1981	1982				
John	189	180	175	Mary	115	120	133				
Mary	115	120	133	Bob	157	158	157				
Bob	157	158	157	Alice	8	25	34				

find

To advance the top line on the screen (or in the frame) to the first line containing a particular string:

WHAT NEXT? **find** *string*

replace

To replace all occurrences of a first string with a second string in the working area:

WHAT NEXT?
replace *string1 string2*

drop

To drop all lines in the working area that contain a particular string:

WHAT NEXT? **drop** *string*

keep

To keep only those lines in the working area that contain a particular string:

WHAT NEXT? **keep** *string*

How to Specify a String

A string is any series of characters. If a string contains no blanks and does not begin with a single or double quotation mark, it can be specified simply by typing it unaltered.

Examples: *alphabet*

4,5
Jone's
&#!\$'"\$.##"'''

If a string contains a blank or starts with a single or double quotation mark, it must itself be enclosed in quotation marks. The enclosing quotation marks (single or double) must not be present anywhere in the string.

Examples: *'NY NY'*
'He said',
"s"

sort

The Sort command organizes the elements of a list alphabetically or numerically in either ascending or descending order. The command sorts lines based on the zone identified by the left-position right-position parameters.

WHAT NEXT? **sort** [options, if any]
left-position right-position

Options

(may select one or none from each category—maximum of three)
When options are not designated, the defaults are: **a c all**

a sorts in ascending order
d sorts in descending order
c sorts characters alphabetically
n sorts numbers
all sorts all lines in working area
s some—sorts only within blocks of consecutive lines that have a plus sign in the first position (useful for tables)

p partial—sorts only within blocks of consecutive lines that have some entry in the zone used as the basis for the sort (useful for subdivided lists)

Example Command: **sort n d 10 20**

Note: To sort on multiple zones, sort first on the least important zone, then the next most important, etc.

Examples of Options

The lines within unshaded areas would be sorted by the option chosen.

ALL

John	31
Bob	21
Bill	52
Mary	25
Jane	26
Rose	51

SOME

ex	Name	Age
+	John	31
+	Bob	21
+	Bill	52
= +	Total	104
+	Mary	25
+	Jane	26
+	Rose	51
= +	Total	102
	= Grand Total	206

PARTIAL

Name	Age
John	31
Bob	21
Bill	52
Mary	25
Jane	26
Rose	51

tally

Tally counts instances of like elements on a list and presents a summary tabulation.

WHAT NEXT? **tally** [*it or disk-file name*] [**all**]
[*key-left-position key-right-position, etc.*]
[**sum** *left-position right-position, etc.*] **end**

Options:

- tally the working file (*it*) or a disk-file (*disk-file name*)
- tally the column (key zone) of one's choice by specifying the position-range of the column to be tabulated
- tally subgroups within groups by specifying multiple key zones.
- sum the values of additional variables by indicating **sum** and the position boundaries of the columns to be summed

Tally normally considers only lines that begin with a plus sign in the first position. If the **all** option is used, then all nonblank lines are tallied. The Tally command is terminated by the word **end**.

The Report

Tally prepares a report which is inserted into the working area. This report has the form of a data file. It can be saved and used directly with the Load command and appropriate masks to prepare summary tables.

The report shows the command used to generate it and the resulting tallies. A result line shows first the values found in the key zones specified. When more than one key zone is used, the values are separated by colons. This is followed by an equal sign and a number representing how many times the combination occurred. Finally, the group totals for any summed variables are presented in the order requested.

Example commands:

tally it all 1 5 end tally survey 1 10 sum 14 20 end tally it 1 10 20 25 end

An Example

Result of: **tally list 8 14 29 36
sum 41 47 19 22 end**

Disk file "LIST"			
Sex	Age	Status	Income
8-14	19-22	29-36	41-47
+	Male	29	Single
+	Female	22	Single
+	Male	31	Married
+	Female	42	Married
+	Male	51	Divorced
+	Male	20	Single
+	Female	25	Single
+	Male	34	Married

TALLY LIST 8 14 29 36 SUM 41 47 19 22 END					
Male	:	Single	=	2	43000
Female	:	Married	=	1	36000
Male	:	Married	=	2	40000
Female	:	Single	=	2	31000
Male	:	Divorced	=	1	42000

load

Load fills out specific zones in the working area with either data from a disk file or with blanks.

WHAT NEXT? **load** [disk-file-name or blank]

In order to use this command, the working area must be a mask, and the disk file must be a data file.

Data Files

A data file consists of lines which assign values to names. To do this, the name of a value is placed on the left side of an equal sign and its value is placed on the right side.

- Lines without an equal sign in a data file are considered to be comments.
- Values containing a space must be enclosed in single or double quotation marks.

Though a line may contain only one name, it may contain more than one value. In such cases, the values must be separated from each other by one or more spaces. The names of successive values are automatically assumed to be "one higher" than the name of the value preceding them.

- For a name ending with an alphabetic character, "one higher" means appending "1" to the end of the name (e.g., **x45y** becomes **x45y1**).
- For a name ending with a number, "one higher" means adding one to that number (e.g., **gnp9** becomes **gnp10**).

Masks

A mask must show both where data should be placed and how data should be presented. To show where data should be placed, braces are used to define the correct zone on a line. To show how data should be presented, the left brace is immediately followed by one of the special characters below.

Character	Interpretation
-	Data centered in zone
>	Data right-justified with right-most limit of zone
<	Data left-justified with left-most limit of zone
!	Data left-justified and unused space after data eliminated ("squeezed")

A name, which corresponds to the name used in the data file, should be put inside the braces to show which data should be placed there. If no name is put inside the braces, the name will be assumed to be one higher than the previous name, just like data files.
(See also "Long Names in Masks")

An Example of Load

A Data File (on disk)

```
Address Information
company = 'Doe & Sons, Inc.'
name = 'John Doe'
street = '1431 West 45th Street'
city = 'New York'
state = 'NY'
zip = '10028'
Sales Figures
sales.80 = 20,000 21,000
```

A Mask (the working area)

```
Name: { <name>
Address: { <street>
           | <city>
           | !state } { <zip>
{ -company
Sales: { <sales.80> }
```

The Result of Load (the new working area)

Name:	John Doe
Address:	1431 West 45th Street
	New York
	NY 10028
Doe & Sons, Inc.	
Sales:	1980 1981
	20,000 21,000

unload

Unload moves (or "unloads") data from the working area to create a new working area which is a data file.

WHAT NEXT? ***unload disk-file-name***

The file on disk must be a mask. *Unload* converts the working area from a text file to a data file on the basis of the mask. The original working area is destroyed in the process.

Masks

When using *Unload*, a mask shows where data are located in the text file and the names of data items to be unloaded. Braces on a line in the mask mean that the corresponding line in the text file should be unloaded. If a line in the mask has no braces, it is ignored.

The lines in the mask must be in the same order as those in the text file. The mask line must match the text line, character for character, for all positions to the left of the first brace on the mask line. This brace can be either an open brace, which would signal the first value to be unloaded, or a close brace, which simply demarks the end of the matching zone.

Long Names in Masks

To enter a data name which is longer than the zone for the data:

{ }	place the braces to define the correct zones for an entire line
{sa}	type the data name for a zone inside the braces until the cursor is over the close brace character
{sa#}	enter insert mode (see Editing Keystrokes)
{sa#les}	type a number sign (#)
	type the rest of the name
	leave insert mode

An Example of Unload

A Mask File (on disk)

This mask unloads name and 1981 Sales

Name: {name }
Sales: } {sales.81 }

A Text File (the working area)

Name: John Doe
Address: 1431 West 45th Street
New York
NY 10028
Doe and Sons, Inc.
Sales: 1980 1981
 20,000 21,000

A Data File (the new working area)

name = 'John Doe'
sales.81 = '21,000'

transfer

The Transfer command transfers values from a file on disk into the working area.

WHAT NEXT? *transfer disk-file-name or blank*

Mask Lines

To use the Transfer command, mask lines are placed in the working area and in the file on disk immediately ahead of the line to which they refer. The shaded area below shows the zone defined as "total" according to the mask line.

{ total }
Total 

Aside from the name of the value (i.e. "total"), the braces must contain a special character. For zones in the working area which are to be replaced by values, the following characters may be used.

- Character Interpretation
- value centered in zone
 - > value right-justified with right-most limit of zone
 - < value left-justified with left-most limit of zone

If none of the above three characters is used, the zone is assumed to be "protected" and no data will be placed into it.

For zones in the disk file which are to be

sources of data, the symbol "!" must be placed within the braces.

A zone may be both a source of data when in a disk file and a destination for data when in the working area (see below).

{ >! ratio }

If a zone in the working area is too small to hold a value, the zone is filled with asterisks.

See: Load (Data Files) for the rules governing the name of a value for a set of braces containing no name. Unload (Long Names in Masks) for the means to specify names which are longer than the zones they define.

Note: Mask lines may be made Print Comment Lines so that they do not appear in printed copy.

The Blank Option

When the word "blank" is given in place of a disk-file-name, all unprotected zones in the working area are filled with spaces.

Example

Disk File "EXP"

Expenses		
ex		999,999.99
+	Rent	1,240.00
+	Food	350.00
=	Total	{ extotal! } 1,590.00

Working Area

Summary		
ex		999,999.00
+	Income	4,000.00
-	Expenses	{extotal>}
=	Net	

Result of: *transfer exp*

Summary		
ex		999,999.99
+	Income	4,000.00
-	Expenses	{extotal>} 1,590.00
=	Net	

addenda

The following apply only to 16-bit versions
of T/Maker

General	A line may be up to 400 column-positions long instead of 300.
Files Command	May be used with ambiguous file names involving " " or "?". Examples: files b: *.bak files a: ??83.tab
Compute Command	An Example Line may contain up to 50 Model Numbers instead of 25. Calculation Line Sequence Numbers may range from zero to 30 instead of zero to 20.
Path Command	The Path Command can be used to change the directory (MSDOS 2.0 or above).
Mix Command	The Mix Command can be used to encrypt or unencrypt the working file based on a password. (Applies to all versions.)

