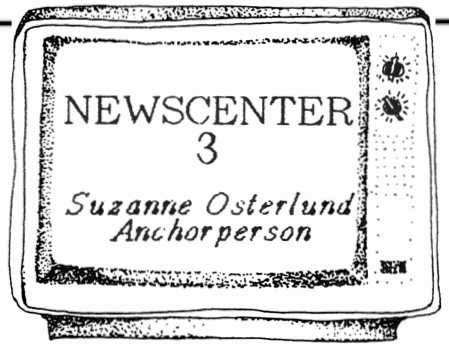
T|MUG

T/MAKER USER'S GROUP NEWSLETTER
VOLUME 2, NUMBER 6, NOVEMBER/DECEMBER 1983

IN THIS ISSUE:



This special issue, our holiday stocking stuffer, includes one of the best applications yet. Besides offering the helpful T/ips section, we present a most informative article about routing information using T/Maker. This feature is written by Ron Roizen who has contributed several valuable articles to T/MUG. "Routing Information With T/Maker", provides a unique yet simple approach to updating several files at once when the need arises to transfer a body of information. Although this system of routing takes some time to learn, it will definitely prove to be a real time-saver. Once you become familiar with this method, you'll be surprised at the convenience and productivity it provides.

Pleasing Press

The September 15th issue of *Computer Shopper*, a Florida based magazine, included a thorough review of T/Maker III. Frank Davidoff in "Software Review -- T/Maker III" writes:

> *...It is a pleasure to review a program with as excellent documentation as T/Maker provides...(p164).*
>
> *...I can generally recommend this program especially for users who have need for word processing and tabular data calculation and manipulation. There are so many capabilities that after getting familiar with it you begin to feel that you can do almost anything to a file...(p165).*

In the September 26 issue of *Computer Systems News*, Michael Azzara interviewed Heidi Roizen about T/Maker III. The article, "T/Maker Unveils Micro-Based Integrated Applications Package", focused on the comparison between T/Maker III and other integrated programs.

The October 10 issue of *InfoWorld* included an insert previewing the magazine's glossy-paged new format. Featured in this insert is a single software review -- a reprint of Tim Daneliuk's July 18 T/Maker III review. This review includes a Software Report Card grading T/Maker III on performance, documentation, ease of use, and error handling. How does T/Maker III make the grade?. . . On all four points -- excellent.

New Distributors

In the last issue, we announced four new distributors of T/Maker software. With this issue are seven more additions to the list: The Future Group (North Carolina), NAPAC (Canada), Nippon Univac Information Systems Kaisha Ltd. (Japan), Oxford/Stanford Corporation (Canada), Redcom S.A. de C.V. (Mexico), Software Control International (New York), and TRP Data Products, Ltd. (Hong Kong).

--NEWSCENTER 3, *continued*

Updates

In the last issue of T/MUG, an announcement appeared asking users of computer terminals not listed by TModify to complete a form and send it to us at T/Maker. The response to the Terminal Specifications form has been good. The offer is still on for users who send in forms for terminals OTHER than the ones listed below. Naturally, T/Maker can only reward tea mugs to the first users who send configurations for unlisted terminals. The Terminal Specifications form is located near the back of this issue.

LEAR SIEGLER - ADM3A
LEAR SIEGLER - ADM31
NORTH STAR ADVANTAGE
RADIO SHACK II - LIFEBOAT CP/M
HAZELTINE 1420, 1500, 1520
KAYPRO II
SOROC IQ 120
ANN ARBOR AMBASSADOR ANSI X3.64
NETRONICS VIDEO TERMINAL
HEATH/ZENITH TERMINAL/COMPUTER
IBM3101
TVI 912/920 TERMINAL
ARCHIVES COMPUTER
PERKIN ELMER - 550
XEROX 820 COMPUTER
OSBORNE COMPUTER
QUT 102
TELEVIDEO 950
AES DATA MODEL 103/LANIER
INTERTEC SUPERBRAIN
ESPRIT I, II

The T/Maker staff led a T/Maker users workshop Saturday, October 15. This first-time event, presented by Computer Sandbox of Sales Research Group, drew a group of eleven users from around the bay area. The five-hour workshop took place in Richmond, California. Users were given detailed instruction in the use of T/Maker III editing and spreadsheet functions. Because of the success of this initial workshop, a follow-up workshop is in the planning stages.

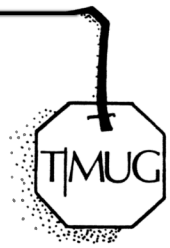
The National Software Show (San Francisco Trade Show Center) of October 19-21 brought in over 300 new contacts for T/Maker Company. An article appeared in the *Gordon Show Dailies*, the show newspaper, which focused upon T/Maker's true integration.

Contributions to T/MUG are always appreciated. If you have discovered a unique way to use T/Maker to simplify your personal or professional life, we want to hear about it. Authors of published articles will receive a handsome T/Maker tea mug. Please address contributions to T/MUG Articles c/o T/Maker Company.

Upcoming Shows

The Las Vegas Convention Center (Las Vegas, Nevada) from November 29-December 2, 1983 will host COMDEX/Fall '83. T/Maker will be exhibiting at the Personal Micro Computers booth, the Sperry booth, and one other to be announced.

T/MUG BACK ISSUES



Several users have requested back issues of the T/MUG Newsletter. They are available in a set (separate back issues cannot be purchased). These issues reflect part of the history of T/Maker Company (before it was called T/Maker Company) as well as the revisions T/Maker has undergone to produce T/Maker III. Here are some highlights:

* 1040 Federal Tax Form * Checking Account System * Analyzing Business Performance * Invoicing System * Opening an IRA * Analyzing Survey Data

The entire set is available for $15 for shipment to US, Canada, and Mexico. Other countries please add $15 for air mail charges, and please submit an international money order or US bank check drawn in US dollars. Make check out to T/Maker Company at 2115 Landings Drive, Mountain View, CA 94043. Please be sure to note it is a back-order so we don't confuse it with a subscription renewal.

Note: This is the last time we will be offering back issues for $15.00. With the new year, the price will go up to $20.00 for shipment to US, Canada, and Mexico, and $40.00 for other countries. Maybe the new price will actually begin to cover newsletter production costs.

T/IPS

SIMPLE FORM LETTER GENERATION:
Quite often it is necessary to send the same letter to a number of different people. It is especially handy to be able to do this to a mailing list which is often re-used. Here is a simple way to generate form letters and re-usable mailing lists.

This method requires two files, one containing the mailing list (MAIL.LST) and the other containing the letter (LETTER).

The mailing list looks something like this:

THIS IS THE FILE "MAIL.LST"

THIS IS THE FIRST TEST OF THE LETTER
The date will begin to appear on the next letter.

.continue letter

King Kong
Empire State Building
New York, NY 10022

Dear Mr. Kong:

.continue letter

Joan of Arc
Arch de Triumph
Paris, FRANCE

Dear Joanie:

.continue letter

Ronald Reagan
The White House
1600 Pennsylvania Avenue
Washington, D.C.

Dear President Reagan:

.continue letter

Notice that you could have easily had more letters with different names, and chosen which letter to send to each person on your list. To do so, you would simply edit the file, and put the name of the particular letter file in the spot after the ".continue" command.

The letter or letters should be similar in form to the following example:

THIS IS THE FILE "LETTER":

<< ->>

Let me just say thank you for your tremendous response to T/Maker Software. We are happy that you have found so many uses for it in your rather unusual profession. Please watch the press for our latest developments.

Thanks again for your support!

Sincerely,

>><<

Heidi Roizen
President

.new

October 7, 1983

Notice two "tricks"in this letter. First of all, the ".new" command moves us to the top of the next letter, on which the date is printed. Second, the period in the first position makes T/Maker insert all those blank lines between where it will print the date, and where it will print the next address it gets from the mailing list. (The period will not print, as it is in effect a blank print design command. It is used solely as a space holder.)

Now, when you GET the MAIL.LST file and PRINT it, you should be on your way.

More T/ips

PRINTING THE LABELS:
You can also turn this mailing list into mailing labels quite easily. First, edit the file and add a PAGESIZE print design command at the top of the file, so T/Maker knows how large your labels are. (The label used to send out the T/MUG newsletter is pagesize 9.)

Next, REPLACE ".continue letter" with ".new". Finally, DROP all lines with the word Dear, to get rid of the salutation. If you have the word Dear in an address (I.E. "Dear Creek Road") you might want to instead DROP lines containing ":" to accomplish the same thing.

Your file now looks like this:

.PAGESIZE 9
.new

King Kong
Empire State Building
New York, NY 10022

.new

Joan of Arc
Arch de Triumph
Paris, FRANCE

. . . and so on.

Just load up your labels, and you are ready to go.

ROUTING INFORMATION WITH T/MAKER

by

Ron Roizen

EDITOR'S NOTE:

The following article offers an invaluable approach to handling the problem of transferring information to several areas quickly and conveniently. Although the method outlined herein takes time to learn, it is efficient, fun, and a true time-saver once it is mastered. This procedure reveals the precision and power of integrated programs at their best. After developing it on disk, you need only enter information once, and the procedure will update all files that are made compatible with it. Your system will practically run itself.

Please set some time aside to go through this set-up and become comfortable with it. Its accelerated pace and efficient processing of data makes it an effective tool that several users can benefit from. After working through the example, the user can create his own personal time-saving system to enhance his work production capabilities. Good luck, and please feel free to send comments, questions, or examples of how this system has helped organize an aspect of business or personal life.

ROUTING INFORMATION WITH T/MAKER

This article deals with a problem that may be familiar to a number of T/Maker users, particularly those in business. It concerns the routing of information to different files--files that for one reason or another must be updated with new information as new information comes in. Suppose, for example, that Sales Manager M is responsible for keeping track of company sales in Widgets, Inc., a company that produces two products, widgets and widget refills. Each time a sale is made three separate files must be updated: the company's monthly record of revenues (File.A), the company's inventory file (File.B), and the individual record of the salesperson who made the sale (this, one of several record files). Sales Manager M wants to automate the process of file updating, so that the appropriate data from each sales event is automatically inserted into each of the three files. How?

Naturally, the particulars of this problem will differ from case to case. In general, though, it presents us with the tasks of sending the right information to the right file and making sure that the information becomes lodged in that file in the right way.

The Program's General Strategy

Before we look in detail at this solution, let us briefly describe the overall strategy it takes. This strategy is premised on the notion that the user should key-in essential sales-event information only once. After that, the program takes over and sends appropriate information to appropriate destinations on its own. As one might guess, the essential T/Maker command for actually transmitting the information from file to file is the LOAD command. In order to use LOAD, of course, we must key-in sales-event information in a format that T/Maker knows as a "Data File" (see the description of LOAD in the Reference Manual). Just to make sure that all the essential information from each sale is actually keyed-in, the program uses a "blank form" file--one that provides a line for each bit of essential sales information. A SAVEd version of this blank form file is then used as the universal source of information for the destination files--it is a sort of "loading dock" ready to provide whatever information other files require.

In order for a destination file to be able to receive information from the "loading dock" file it must have a properly positioned mask in it. Different destination files will want different pieces of information from the loading dock file and so they will require different masks. For this reason, this routing program involves the creation of a number of "mask files"--these contain nothing else but the appropriate masks for one or another

destination file.

How, then, does the program work? It's simple! It first accepts and SAVEs the essential information from a given sales event--making the "loading dock" file. Then, it GETs a destination file into memory. Then, it INSERTS the correct mask into the right place in that file. Then, it LOADS the "loading dock" file, which places the appropriate data into the destination file. Then, it COMPUTEs and SAVEs the updated destination file.

Want that a little slower? In what follows these various operations are described in detail.

The Blank Form File

First of all, we'll need a file that is essentially the equivalent of a reusable blank form, a form used for filling in the necessary sales information from each sales event. Let's name this file "Blank.frm". Blank.frm might look something like this:

[THE BLANK.FRM FILE]

```
Salesperson          =
Salesperson.id       =
Date                 =
Widget.quan          =
Refill.quan          =
Sale.$               =
Com.r                =
```

This file should contain all of the information we are going to require in any of the three destination files. Notice, by the way, that we have included a "Salesperson.id" code--this might be the initials of the salesperson. ("Com.r" stands for the commission rate the salesperson earned on the sale.) The Blank.frm file will become the original source of all the data to be placed in other files. Later on, we will be LOADing this information into Files .A, .B, and record files of individual salespeople.

But there is a dilemma here! For Blank.frm to become a general source of information it will have to be SAVEd, and SAVEd after the information from a given sale has been filled in. But were we to key-in a given sale's information and then SAVE the file, then our blank form file would no longer be blank for future uses! Consequently, the first thing our T/Maker program must do is make a "saved version" of the Blank.frm file--that is, a version under a new name and one that is SAVEd after the sales data have been keyed in. This is accomplished by adding a DO command to the top of the Blank.frm file, like this:

[THE BLANK.FRM FILE]

```
DELETE Loading.dok RENAME Loading.dok SAVE


Salesperson          =
Salesperson.id       =
Date                 =
Widget.quan          =
Refill.quan          =
Sale.$               =
Com.r                =
```

Now, after invoking a DO command, we have a file named Loading.dok on disk that can serve as the source of information for the LOADs that will update destination files. What next?

Updating File.A

Suppose that the company's revenues record file--File.A--looked something like this:

[FILE.A]

MONTHLY RECORD OF REVENUES:

	[Col 1] Salesperson Code	[Col 2] Com. Rate	[Col 3] Date	[Col 4] Amount	[Col 5] Net to Co.
ex		9.99		9999.99	999,999.99
uc1	sta+sfo+fta			*	=
zv					
+	jh	0.30	8-20	49.99	34.99
+	hr	0.10	8-20	57.50	51.75
+	rt	0.10	8-21	150.75	135.68
jc2		ftz			
=	Total			258.24	222.42

File.A keeps a running record of the Widget Company's monthly revenues. We see that only four of the seven pieces of information originally keyed-in on the Blank.frm file are actually used in File.A: the salesperson code (in Col 1), the salesperson's commission rate (in Col 2), the date of the sale (in Col 3), and the total amount of the sale (in Col 4). From these four facts File.A calculates the net revenues to the company from each sale (in Col 5) and two overall totals: total sales revenues ($258.24) and total net revenues after the sales commissions have been subtracted ($222.42).*

If we were going to update this file from the keyboard (using the Blank.frm/Loading.dok files already created) we would inscribe

a mask line into File.A just below the last sales event. The mask line would look like the line we have underscored:

[FILE A]

```
|
|   MONTHLY RECORD OF SALES:
|
|   Salesperson Code      Com. Rate    Date    Amount    Net to Co.
|   ----------------      ---------    ------  -------   ----------
|ex                          9.99                9999.99  999,999.99
|uc1                  sta+sfo+fta                   *          =
|zv
|+        jh                 0.30      8-20      49.99       34.99
|+        hr                 0.10      8-20      57.50       51.75
|+        rt                 0.10      8-21     150.75      135.68
|+       {<Salesperson.id}{<Com.r}   {<Date}{<Sale.$}
|jc2                         ftz
|=        Total                                 258.24      222.42
|
```

To update the file, now, we would simply LOAD the Loading.dok file, move the cursor to the top of File.A, quit the editor, invoke the COMPUTE command, and finally SAVE the updated file. Mission accomplished!

The procedure for automatically updating File.A is not very much different. It requires a new file though. Let's name it Mask.a. Mask.a consists of a single line--namely, the mask-line that we just employed in File.A's manual update:

---

* The actual calculations in File.A are performed as follows: The "uc1" Horizontal Calculation Line (HCL) first stores the commission rate to be paid the salesperson. Then, it enters that rate into the T/Maker calculator (with the first plus sign) and subtracts the rate from one (with the sfo terminator symbol). Next, it enters the result into the T/Maker calculator (with the second plus sign) and retrieves the original salesperson commission rate for future use (with the fta). The sales commission rate subtracted from one represents, of course, the proportion of the sales revenue that will go to the company. This proportion is then multiplied by the sale price (by the *), and the result is produced in the final column. The zero values line makes exactly-zero values "print" as blanks. And, finally, the jc2 HCL uses a little T/Maker trick to suppress the presentation of a meaningless value: As one can see, the sum of the commission rates (in column two) is a meaningless value. Its presentation is suppressed by fetching "z". Since "z" was never stored in this calculation, the value fetched is zero. And since the zero-values line prints exactly zero values as blanks, that value is suppressed.

[THE MASK.A FILE]

```
|                                                                              |
|+        {<Salesperson.id}{<Com.r}   {<Date}{<Sale.$}                         |
|______________________________________________________________________________|
```

Once the Mask.a file has been created, we can automate the updating of File.A with a single string of commands. Namely:

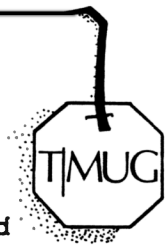
```
WHAT NEXT? GET FILE.A FIND jc2 INSERT Mask.a 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE
```

What does this command string do? First, it gets File.A into the memory. Next, it sends the cursor to the line on which the "jc2" control code appears. Next, it inserts the Mask.a file--remember that an INSERTed file is placed **above** the line on which the cursor lies at the moment the INSERT command is invoked. Thus, the mask will be placed at the right spot for each successive sales event. Next, it sends the cursor back to the top of the file (with the "1/1" command). Next, it LOADs the required sales information from the Loading.dok file. Next, it LOADs blanks for any and all information missing from the Loading.dok file--this, just in case some information happens to be missing. Next, it COMPUTEs the new table, and, finally, it SAVES the updated file.

As it happens, the best place to lodge this group of commands is on the Blank.frm file. It becomes the second "DO-line" in that file. Also note that we must add the word "DO" to the end of the first DO-line in that file, so that T/Maker will know to continue on to the second DO-line.

[THE BLANK.FRM FILE]

```
|DELETE Loading.dok RENAME Loading.dok SAVE DO                                 |
|GET FILE.A FIND jc2 INSERT Mask.a 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE|
|                                                                              |
|                                                                              |
|    Salesperson        =                                                      |
|    Salesperson.id     =                                                      |
|    Date               =                                                      |
|    Widget.quan        =                                                      |
|    Refill.quan        =                                                      |
|    Sale.$             =                                                      |
|    Com.r              =                                                      |
|______________________________________________________________________________|
```



Updating File.B

Suppose the company's inventory file-- File.B --looked something like this:

[FILE.B]

```
|                                                                          |
|   WIDGET INVENTORY RECORD:                                               |
|                                                                          |
|            [Col 1]       [Col 2]   [Col 3]        [Col 4]               |
|           Salesperson      Date  Widget Quan.   Refill Quan.             |
|           -----------     ----   ------------  ------------             |
|ex                                      9,999         9,999               |
|zv                                                                        |
|+          INITIAL STOCK:                350           981               |
|-          Jeff Hall         8-20          3             0               |
|-          Harry Renner      8-20          5            52               |
|-          Rena Thompson     8-21         11            20               |
|=          TOTAL LEFT:                   331           909               |
|__________________________________________________________________________|
```

File.B uses only four pieces of information drawn from the Blank.frm/Loading.dok files: the salesperson's name (in Col 1), the date (in Col 2), the quantity of widgets and the quantity of refills sold (in Cols 3 and 4). This file employs a single Vertical Calculation Strip which simply subtracts quantities sold from stock on hand.

Once more, we will need to create a mask file appropriate for use in File.B--let's call it "Mask.b":

[THE MASK.B FILE]

```
|                                                                          |
|-          {<Salesperson}    {<Date}   {<Widget.quan}{<Refill.quan}       |
|__________________________________________________________________________|
```

And once more a single command string will update the inventory file:

```
WHAT NEXT? GET FILE.B FIND TOTAL INSERT Mask.b 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE
```

And, as before, the best place for this group of commands is the Blank.frm file. Notice that we have added "GET Blank.frm 3/1 DO" to the second DO-command line--this in order to send T/Maker to the third DO-command line after File.A has been updated.

[THE BLANK.FRM FILE]

```
!DELETE Loading.dok RENAME Loading.dok SAVE DO
!GET FILE.A FIND jc2 INSERT Mask.a 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Blank.frm 3/1 DO
!GET FILE.B FIND TOTAL INSERT Mask.b 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE
|
|
|    Salesperson        =
|    Salesperson.id     =
|    Date               =
|    Widget.quan        =
|    Refill.quan        =
|    Sale.$             =
|    Com.r              =
|
```

Updating the Individual Salesperson's File

Finally, we want to update the file of the person who made the sale. Suppose that each salesperson in the firm has a file that looks like this:

[Model Salesperson File]

RECORD OF SALES FOR RENA THOMPSON:

	[Col 1] Salesperson	[Col 2] Date	[Col 3] Com. Rate	[Col 4] Sale Amt.	[Col 5] Tot. Due
ex			9.99	9,999.99	9,999.99
zv					
uc1			+	*	=rnd
+	Rena Thompson	8-17	0.30	577.85	173.36
+	Rena Thompson	8-19	0.30	49.99	15.00
+	Rena Thompson	8-21	0.35	1,891.80	662.13
jc2			ftz		
=	TOTAL			2,519.64	850.49

The procedure for updating this file involves much the same procedure as we've already seen for the revenue and inventory files. We shall have to create an appropriate mask file (Mask.c), INSERT it, LOAD it, COMPUTE and SAVE it.

The Mask.c file will be this:

[THE MASK.C FILE]

```
|
|+          {<Salesperson}    {<Date}   {<Com.r} {<Sale.$}          |
|_____________________________________________________________________|
```

There is a new wrinkle in the routing problem when we turn to the salesperson's file, however. In the case of the revenue and inventory files it did not matter which salesperson made the sale--no matter which one, Files .A and .B had to be updated. In the case of the individual salesperson, however, the program needs to be informed which salesperson made the sale. It must update the file of only the correct salesperson.

Suppose that the Widget Company employs ten salespeople. Each one has an individual record file like the one shown above. Let's suppose, further, that these files are individually named by using the salesperson's initials followed by ".rec" (for "record").

Salesperson's Name	Salesperson's Record-File's Name
Jeff Hall	jh.rec
Harry Renner	hr.rec
Rena Thompson	rt.rec

How, then, to tell the program which salesperson made the sale? We will accomplish this end by LOADing the salesperson's identifying initials into a command string. The command string looks like this:

```
WHAT NEXT? GET {!Salesperson.id}.rec FIND jc2 INSERT Mask.c 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE
```

Notice that we have used the "!" option in the salesperson mask in order to join the ".rec" to initials in the salesperson's id. Now, let us lodge this command line in the Blank.frm file:

[THE BLANK.FRM FILE]

```
!DELETE Loading.dok RENAME Loading.dok SAVE DO
!GET FILE.A FIND jc2 INSERT Mask.a 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Blank.frm 3/1 DO
!GET FILE.B FIND TOTAL INSERT Mask.b 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE
!GET {!Salesperson.id}.rec FIND jc2 INSERT Mask.c 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE
|
|
|    Salesperson        =
|    Salesperson.id     =
|    Date               =
|    Widget.quan        =
|    Refill.quan        =
|    Sale.$             =
|    Com.r              =
|
```

Notice, now, that the Blank.frm file includes a data line that holds the Salesperson's id or initials, the second data line. Thus, we can "inform" the program of the salesperson who made the sale by LOADing "Loading.dok" (the saved version of Blank.frm) to itself. The necessary commands are added to the first command line of the Blank.frm file.

[THE BLANK.FRM FILE]

```
!DELETE Loading.dok RENAME Loading.dok SAVE LOAD Loading.dok SAVE DO
!GET FILE.A FIND jc2 INSERT Mask.a 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Blank.frm 3/1 DO
!GET FILE.B FIND TOTAL INSERT Mask.b 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE
!GET {!Salesperson.id}.rec FIND jc2 INSERT Mask.c 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE
|
|
|    Salesperson        =
|    Salesperson.id     =
|    Date               =
|    Widget.quan        =
|    Refill.quan        =
|    Sale.$             =
|    Com.r              =
|
```

The first command line will now (1) RENAME Blank.frm to Loading.dok, (2) SAVE Loading.dok, (3) LOAD the Loading.dok file now saved on disk into the Loading.dok presently in memory, and, finally, (4) SAVE the newly-loaded Loading.dok. In all, this will transform the fourth command line into one that "knows" which salesperson made the sale.

Two final additions need to be made to these command lines. (1) After our program has finished processing File.B (on the third command line) it must be told to find the command line directing

it to the Salesperson's individual file. This is accomplished by adding "GET Loading.dok 3/1 DO" to the third command line (this has been shown in bold below). This will send T/Maker to a command line with the salesperson's id filled-in. (2) We shall end the program by sending T/Maker back to a fresh Blank.frm file. This, so that the information from a new sale can be keyed in. "GET Blank.frm" has been added (in bold) to the end of the last command line. Thus, our finished Blank.frm file looks like this:

[THE FINISHED BLANK.FRM FILE]

```
|DELETE Loading.dok RENAME Loading.dok SAVE LOAD Loading.dok SAVE DO                    |
|GET FILE.A FIND jc2 INSERT Mask.a 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Blank.frm 3/1 DO |
|GET FILE.B FIND TOTAL INSERT Mask.b 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Loading.dok 3/1 DO |
|GET {!Salesperson.id}.rec FIND jc2 INSERT Mask.c 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Blank.frm |
|                                                                                       |
|                                                                                       |
|     Salesperson          =                                                            |
|     Salesperson.id       =                                                            |
|     Date                 =                                                            |
|     Widget.quan          =                                                            |
|     Refill.quan          =                                                            |
|     Sale.$               =                                                            |
|     Com.r                =                                                            |
|_______________________________________________________________________________________|
```

Running the Routing Program

Running the routing program is very easy to do. All the user must do is (1) GET the Blank.frm file; (2) fill in the information from a given sales event (as has been done below--note, incidentally, the use of quotation marks for the salesperson's name); (3) move the cursor to the top of the file; and, finally, invoke the DO command. The program will now chug along updating Files .A and .B and the file of the right salesperson!

```
!DELETE Loading.dok RENAME Loading.dok SAVE LOAD Loading.dok SAVE DO
!GET FILE.A FIND jc2 INSERT Mask.a 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Blank.frm 3/1 DO
!GET FILE.B FIND TOTAL INSERT Mask.b 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Loading.dok 3/1 DO
!GET {!Salesperson.id}.rec FIND jc2 INSERT Mask.c 1/1 LOAD Loading.dok LOAD blank COMPUTE SAVE GET Blank.frm
|
|
|   Salesperson      =   "Rena Thompson"
|   Salesperson.id   =   rt
|   Date             =   8-21
|   Widget.quan      =   4
|   Refill.quan      =   5
|   Sale.$           =   792.11
|   Com.r            =   .30
|
```

Summing Up

A great many modifications and refinements might be introduced into this sort of program, and, of course, the calculations associated with revenues, inventory, and salespeople might be made much more elaborate and complex. The point of this article, however, is merely to show a technique for routing new information to a number of destination files. As we have seen, that technique involves the creation of a Blank.frm file, a Loading.dok file (which, in fact, is created by the program itself) and a series of tailored "mask" files appropriately designed for each of the files to be updated. The commands that run this T/Maker routing program, as we have seen, lie in four command lines at the top of the Blank.frm file.

WHERE TO BUY T/MAKER III (OR UPGRADE FROM T/MAKER II)

DOMESTIC DISTRIBUTORS

COMPUTER POTENTIALS, 2350A Walsh Avenue, Santa Clara, CA 95051,
(408) 980-9100, 800-457-4177.
DATASOLVE CORPORATION, 727 N. Hudson, Chicago, IL 60610, (312) 943-9141.
THE FUTURE GROUP, 212 S. Tryon, Suite 1270, Charlotte, NC 28202,
(704) 334-3001.
LIFEBOAT ASSOCIATES, 1651 Third Avenue, New York, NY 10028,
(212) 860-0300.
MICROHOUSE, 1444 Linden Street, Bethlehem, PA 18016, (215) 868-8219,
800-523-9511.
SOFTSEL COMPUTER PRODUCTS, 546 N. Oak Street, Inglewood, CA 90302,
(213) 412-1700, 800-645-7777.
SOFTWARE CONTROL INTERNATIONAL, 91 N. Franklin St., Hempstead, NY 11550,
(516) 538-5300.
SOFTWARE WHOLESALERS, 31 Memorial Drive, Avon, MA 02322,
(617) 587-2904, 800-633-1000.
WESTICO, 25 Van Zant Street, Norwalk, CT 06855, (203) 853-6880.

INTERNATIONAL DISTRIBUTORS

ALFATRON PTY. LTD., 1761 Ferntree Gully Rd., Ferntree Gully,
VIC 3156, Australia.
AUSDATA, P.O.Box 664, Chatswood, N.S.W. 2067, Australia.
COMPUTER TECHNIQUES, Atherstrasse 5, Postfach 781, CH 6301 Zug, Switzerland.
LIFEBOAT ASSOCIATES LTD., P.O.Box 125, London WC2H 9LU, England.
LIFEBOAT FRANCE, 70 Avenue d/Argenteuil, 92600 Asnieres, France.
LIFEBOAT SCANDINAVIA, Per Nilsabyn 46, Z4500 Staffanstorp, Sweden.
LIFEBOAT SWITZERLAND, Hinterbergstrasse 9, CH-6330 Cham, Switzerland.
M & T SOFTWARE VERLAG, Hans-Pincel Strasse 2, 8013 Haar bei Munich,
West Germany.
NAPAC SOFTWARE, 370 Donald St., Suite 100, Winnipeg, Manitoba, Canada.
NIPPON UNIVAC INFORMATION SYSTEMS KAISHA LTD., 17-22, 2-chome, Akasaka,
Minato-ku, Tokyo, Japan.
OXFORD/STANFORD CORP., 6869 Seller Avenue, Burnaby, British Columbia,
V5J 4R2 Canada.
REDCOM. S.A. de C.V., Blvd. Manuel Avila Camacho, No. 95-1er. Piso,
53000 San Bartolo Naucalpan, Edo. De Mexico.
TRP DATA PRODUCTS LTD., 25 Chatham Road South, Kowloon, Hong Kong.

T/Maker Terminal Specifications:

			Example ADDS Viewpoint	Your Terminal Computer
I.	**ROW/COLUMN:**			
1.	Number of Rows		24	_____
2.	Number of Columns		80	_____
II.	**ADDRESSING THE CURSOR:**			
	When addressing the cursor, will values be:			
3.	Single Characters or Character Digits? (usually row or column number should be sent as a single character corresponding to the number (eg. 45 for -) and not the two characters 4 and then 5.)		Single Chars	_____
	T/Maker needs the sequence for absolute cursor addressing.			
	How do you address the:			
4.	Top Row (usually 0 or 32)	Space	32	_____
5.	Leftmost Column (0 or 32)	Space	32	_____
	(eg. top left corner is (32,32) ASCII(Space,Space)			
	What is the sequence to position the cursor:			
6.	Number 1	ESC	27	_____
7.	Number 2	Y	89	_____
8.	Number 3	row	255	_____
9.	Number 4	column	254	_____
10.	Number 5		0	_____
11.	Number 6		0	_____
12.	Number 7		0	_____
13.	Number 8		0	_____
	(eg. to position cursor at screen center (12,40) sequence is : "ESC, Y, +, G" for ADDS terminal)			
III.	**MANDATORY FEATURES:**			
	These are the minimum terminal features required to run T/Maker:			
14.	Carriage Return	^M	13 0	_____
15.	Line Feed	^J	10 0	_____
16.	Backspace	^U	21 0	_____
17.	Clear Screen	^L	12 0	_____

....continued next page.....

IV. OPTIONAL FEATURES:

These features enhance and speed up the operation of the program. They are not required but are desirable.

18.	Make Beep	^G	7	Ø	_____
19.	Erase to End of Line		Ø	Ø	_____
20.	Erase to End of Screen		Ø	Ø	_____
21.	Line Insert		Ø	Ø	_____
22.	Line Delete		Ø	Ø	_____
23.	Character Insert		Ø	Ø	_____
24.	Character Delete		Ø	Ø	_____