T|MUG

T/MAKER USER'S GROUP NEWSLETTER
VOLUME 4, NUMBER 4, JULY/AUGUST 1985

In This Issue:

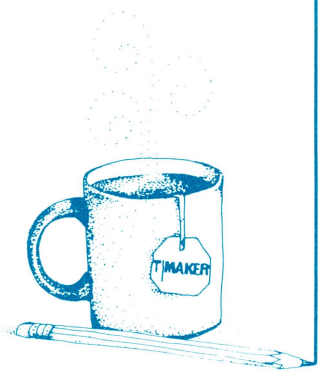
T/Maker's Database Made Easy 3

DB Method for Giving Commands 10

The T/Maker Typewriter 15

Sales Forecasting 16

Dressing Up Input Screens 19



T/Maker Users' Group Newsletter

T/MAKER NEWSFRONT

-- Heidi Roizen

What Industry Slowdown?

Not to brag too much -- and not to tempt fate -- but we wanted to let you know that T/Maker just ended its second year as a corporation with some pretty decent results. Since we are completely privately held, we've never disclosed the gory details, but everyone here was pleased to see revenues up over 100% (we're somewhere in the $1.5 million neighborhood) and net income up about 150%.

We wanted to thank all you T/Maker users out there for helping us prove that there is still room for smaller software companies to succeed.

German Version Nears Ship Date

By the time you read this, the German version of T/Maker will be at the press. We're hoping that by adding German to our range, we can increase sales in the European market. Much thanks to Emil Widmer of CFM, Zug, Switzerland and Peter Wicht of GNOSIS, Seeheim-Jugenheim, West Germany for their expert assistance.

Visit us in Boston

Pretty much the whole T/Maker crew will be at the MacWorld Expo in Boston August 21-23. Come by our booth and see our *other* products. We will have a few complimentary tickets, so if you want one send a self-addressed, stamped envelope and we'll send them until we run out.

T/Maker Increases Staff by 17% On Same Day as Apple Layoffs

In an effort to ease the slackening job market in Silicon Valley and facilitate a major reorganization (of the supplies room, stock room, literature racks, etc.) T/Maker Company increased its staff by 17% in one day by adding Ted Schlein to the summer roster. Ted will likewise reduce the staff by about the same percent when he returns to the University of Pennsylvania for his senior year. Ted's duties have included tech support, software bartering, and many other creative functions he will probably never have to duplicate in any other company again. Welcome aboard, Ted!

Greatest Trinket Ever for T/Mug Contributors

We are still, as always, seeking interesting applications for the T/Mug. As usual, we are offering in exchange for your articles the latest T/Maker trinket. This time we have a couple of extra staff teeshirts. There are only 40 of these in the entire world and most of them are in our hands or on the backs of our loved ones. So please, contribute your article on any disk format (accompanied by a print-out) and become one of the select few with a T/Shirt! Articles can be on any topic, simple or complex, specific or generic.

ANNOUNCING T/BILL!

T/Bill is a new, automated system -- written in T/Maker -- for handling accounts receivable or monthly billing. It lets you:

1) Log and calculate all current sales and payments.

2) Archive past sales and payment experience.

3) Prepare a monthly bill for all clients whose accounts were active during the preceding month or who show past-due balances.

   The bill T/BILL prepares includes: (a) the customer's name and address, (b) a line entry for each sale and payment in the preceding month (ordered by date), (c) a line entry for each past month with a past-due balance, and (d) the new current balance.

T/BILL is priced at $125.00 plus a variable fee ($50 to $100) for customization to your reauirements. The system includes complete reference documentaiton. Please contact:

Ron Roizen
1818 Hearst Street
Berkeley, CA 94703
(415) 848-9098

T/MUG PRICES FOR SOFTWARE:

The following are prices for various T/Maker options that we often receive requests for. Orders can be placed through us and we'll take them over the phone for COD, VISA, or MasterCard payments. California residents add 7% sales tax. Shipping is $6.50 for UPS blue and $3.13 for UPS ground. And remember, we MUST know what kind of computer and operating system you have! Our number is (415) 962-0195.

UPGRADE FROM T/MAKER III TO T/MAKER INTEGRATED SOFTWARE MUST HAVE PROOF OF T/MAKER III SERIAL NUMBER:	
FOR SAME COMPUTER TYPE: WILL RECEIVE BOUND MANUAL	$175.00
FOR OPERATING SYSTEM OR COMPUTER OTHER THAN CURRENT LICENSE: WILL RECEIVE BOUND MANUAL	$250.00
T/MAKER INTEGRATED SOFTWARE FOR SECOND OPERATING SYSTEM BOUND MANUAL. MUST HAVE PROOF OF T/MAKER I.S. SERIAL NUMBER.	$225.00

T/MAKER'S DATABASE MADE EASY (EASIER?)

-- Royal Farros

Editor's note: Ever since we added the database to T/Maker, we have been finding more and more applications for it. Its unique structure makes it a perfect fit with the rest of T/Maker -- as well as making it very flexible and powerful.

However, making the most of the database requires some fundamental understanding of what is going on in it. This article was written as a result of numerous customer questions and experiences. We hope it helps you get more out of (or at least, more into) your databases.

A DATABASE is nothing more than a structured way of storing information, and is very similar to a filing cabinet.

We can, however, do a sloppy job setting up our filing system, limiting the amount of information we hold as well as its accessibility.

The next few pages talk about the correct way to set up our filing cabinet (our database). We'll also talk about how we can get to information quickly, how we can print out this information easily, and what to do if our filing cabinet is too small for our current needs.

ANATOMY OF A DATABASE: THE BASICS

A T/Maker database has two parts:

1) A *Form Definition* that tells T/Maker how to **present** or **view** information on the screen.

2) A *Record Definition* that tells T/Maker how to **store** information.

In order to optimize our storage space, we will want to make the Record Definition as small as possible. The Form Definition, however, can be as "fancy" as we wish since it has nothing to do with the way T/Maker will store information.

On the following page is an example of an "efficient" database.

```
<FORM>

   ........................... ADDRESS CARD ............................

 Last Name: {LASTNAME }   FirstName: {FIRSTNAME}

 Address: {ADDRESS              }

 City: {CITY                  }     State: {STATE}    Zip Code: {ZIP}

  ..................................................................

<END>


<RECORD>
01 {LASTNAME }
02 {FIRSTNAME}
03 {ADDRESS              }
04 {CITY                 }
05 {STATE}
06 {ZIP}
<END>
```

Please note a few things about the above example.

* The FORM DEFINITION is the fancy "entry" screen you will see when using the command Update to enter information.

  Notice we can place our information FIELDS anywhere on a SINGLE screen, and use as much "designer" text as we wish.

* The RECORD DEFINITION has NO BLANK LINES.

  Each FIELD is on a separate line, starting with a unique, two digit number.

* The FIELDS in the FORM and RECORD DEFINITIONS are identical in terms of size, spelling, and capitalization.

PRINTING DATABASE REPORTS WITH T/MAKER

Storing records - and printing records - are two different T/Maker activities. If you want your database report to look like your Form Definition, the easiest way to do this

is to:

1) What Next? GET ADDRESS (or whatever your file is called.)

2) What Next? RENAME ADDRESS.RP

3) Jump into EDIT mode and placed cursor on the line with <FORM> on it. Type <RECORD> over <FORM>.

4) Insert a line under the <RECORD> line and type: .NEWPAGE

5) Jump down to the first <END>. Exit the Editor and type at WHAT NEXT? the command: CLIP AFTER

6) To save this as a blank report form, type: SAVE

7) Next, type the command at WHAT NEXT? SELECT ADDRESS END

8) Finally, to print this report, type: PRINT FROM 3 IT (Meaning start printing from the 3 page on - remember, our Definition occupies 2 pages in this set up.)

Below is what your new ADDRESS.RP should look like.

```
<RECORD>
   .NEWPAGE


 Last Name: {LASTNAME }    FirstName: {FIRSTNAME}

 Address: {ADDRESS              }

 City: {CITY                  }     State: {STATE}     Zip Code: {ZIP}

<END>
```

Please note:

* Notice that it's okay to have BLANK LINES in a report-database, because we aren't planning to do anything but print with it. This file would "blow up" if we tried to do anything else with it (like using the SET or ORDER command).

---

* If you run out of space, keep reading. The section on "Creating Really Large Databases" will tell you how to get around space problems.

PRINTING A UNIQUE REPORT:

Suppose we wanted to print out a single line report of our ADDRESS CARD example above. The easiest way to do this is to:

1) What Next? GET ADDRESS (or whatever your file is called.)

2) What Next? RENAME PHONEBOOK.RP

3) Jump into EDIT mode and placed cursor on the line with <RECORD> on it. Jump out of the Editor and type at What Next? prompt: CLIP BEFORE EDIT

4) Jump down to the <END>. Exit the Editor and type at WHAT NEXT? the command: CLIP AFTER 1 EDIT

5) Erase the numbers (field labels) in front of each field.

6) Jump to the end of the {LASTNAME } line and use the JOIN LINE keystrokes (ESC-F10 or ESC <). Use the end-of-line keystroke again, and join lines again. Do this until all the fields on the same line.

7) To save this as a blank report form, type: SAVE

8) Next, type the command at WHAT NEXT? SELECT ADDRESS END

9) Finally, to print this report, type: PRINT FROM 3 IT (Meaning start printing from the 3 page on - remember, our Definition occupies 2 pages in this set up.)

Below is what your new PHONEBOOK.RP should look like.

```
<RECORD>
{LASTNAME },{FIRSTNAME}{ADDRESS              }{CITY                  }{STATE
<END>
   .NEWPAGE

      Name                  Address                  City          State
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      ..<Here>
```

Please note:

---

* Do all of your Ordering or Sorting BEFORE selecting into the Phonebook.rp report database. (You could, however, place a field label like a "*" in front the fields. This would make this a valid database, one suitable for Ordering or manipulating. It might not be very efficient, though.)

* We didn't have to use all the same fields. We could have left some out, and even included some new ones.

The important thing is, if we want to move information from a file on disk to a file on the screen, the field name where that piece of information lies must be identical in both database files.

CREATING REALLY LARGE DATABASES:
LINKING SIMILAR DATABASES WITH <CONTINUE>

Creating a really large database may take creating many smaller identical databases and "linking" them together.

You can "link" identical databases together with the <CONTINUE> filename option.

For example, suppose we wanted to link two files - FILE.A and FILE.B.

FILE.A would look like:

```
<Record>
01 {NAME}
<End>
01 Bob
01 Tom
<continue> FILE.B
```

FILE.B would look like:

```
<Record>
01 {NAME}
<End>
```

Please notice a few things about the above example.

* Database Record Definitions must be identical.

* The last line of each file will contain <CONTINUE> and the name of database that is next in line.

To ensure that the Record definitions in the above example were identical, I did the following:

1) What Next? GET FILE.A

2) What Next? RENAME FILE.B

3) Jumped into EDIT mode and placed cursor on the line with <END> on it. Exited the Editor and typed at WHAT NEXT? the command: CLIP AFTER

4) What Next? SAVE

HINTS FOR ORGANIZING LOTS OF SMALLER DATABASES: CATEGORIZE

It is very important to "categorize" or modulize your linked databases. That is, have some type of overriding organization to your system. Good organization will help us get at information very quickly.

At T/Maker, we place all the orders for a particular month in a database called JAN.DB - which is linked to FEB.DB - which is linked to MAR.DB - and so on.

You might want to alphabetically categorize your databases. For example, file AthruL.DB contains all the names A thru L, and MthruZ.DB contains names M thru Z.

If we have done a good job at organizing our databases, we will be able to retrieve information quickly by going right to the appropriate database.

FINDING A PARTICULAR RECORD IN A SERIES OF LINKED DATABASES

Suppose we've organized our databases by month, and a customer calls and wants to check on an order - but doesn't remember what month the order was placed in.

We can use the SELECT command to extract or retrieve particular information from the linked databases.

Let's select information about the customer Flinstone.

1/1 What next? GET JAN.DB SELECT JAN.DB WHEN CUSTOMER = Flinstone END

A couple of important things to note:

* We used "SELECT JAN.DB" instead of the normal "SELECT IT". This tells T/Maker to look beyond the current file into the linked or continued databases.

* SELECT "cuts out" the un-SELECTED records from current work space. Therefore, never do a SAVE after a SELECT unless you RENAME the current work space first (else you will lose the "temporarily" discarded Records).

FINDING A LOT OF RECORDS IN A SERIES OF LINKED DATABASES:

When you SELECT across databases, the resulting database or report MUST fit in the work file. If you still need more space, try SELECTing in blocks, using the "FROM..TO.."option in the SELECT command. Here's an example:

1/1 what next? SELECT JAN.DB from 1 to 500 when ZIP = 90010 END

1/1 what next? SELECT JAN.DB from 501 to 1000 when ZIP = 90010 END

OTHER QUICK TIPS

FIELD NAMES and CAPITALIZATION ("...FIELD NOT IN RECORD")

Ever see the T/Maker error message, "...FIELD NOT IN RECORD"? You may be spelling the field name correctly, and still, T/Maker refuses to help you locate the desired field.

The problem is more than likely capitalization. While T/Maker doesn't care if commands are capitalized or not, it is VERY picky about the capitalization of field names.

For example, if you defined the field {CITY} in a Record & Form Definition, and then try to "SELECT IT WHEN City = SF END" - T/Maker won't know what you are talking about because "CITY" and "City" are two different names.

Capitalization is also important when finding information in a database.

For example, if you enter a name in a database as "WILMA" - and then search for "Wilma" - T/Maker will beep at you telling you that it can't find "Wilma" - because T/Maker only knows "WILMA". Remember, "Wilma" and "WILMA" are two different people to T/Maker.

One last tip on searching a database. Always start searching from the top of a database. T/Maker always looks forward in a database - so if you are on record 50 of a 100 record database, T/Maker will only search in the last 50 records for a name if you don't HOME first.

NUMBER OF RECORDS IN A DATABASE

A quick way to find the number of records you have in a database is to SELECT the entire database into itself.

For example, by typing at WHAT NEXT? SELECT IT END - you should see a message flash "100 RECORDS SELECTED". That number is the number of records you have in your current database.

A "Database Method" For Giving T/Maker Commands

-- Ron Roizen

There are a variety of ways to command T/Maker: (1) you can use ordinary commands in response to T/Maker's "WHAT NEXT?" prompt, (2) you can write DO Command Lines at the tops of files, and (3) you can even invent your own T/Maker commands by borrowing from the HELP.TMK file.

This article describes yet another approach to commands, one that might be called the "Database Method." Ordinarily, one uses T/Maker's database system to store and manipulate bodies of data. The database system, however, has properties that make it useful for special command needs. In particular, it provides a convenient device for setting up work for relatively inexperienced T/Maker users.

The Database Command Method: How It Works

Let me illustrate the principle with a very simple example. Suppose you wanted to GET and EDIT a file from disk. How would the database command method go about doing this?

Create a file called WORK.1 that looks like this:

THE FILE WORK.1 :

```
18 do
<form>


  1.  Please fill in the name of the file you wish to GET and
      to EDIT below.

                          FILE'S NAME:  {file.n}

  2.  Then, QUIT the UPDATE mode and DO.

<end>
<record>
get {file.n} edit
<end>
```

Take a good look at this file.

WORK.1's top line consists of a simple DO Command Line, one that instructs T/Maker to go to the 18th line of the file (that is, the 18th line once the DO line itself has been removed) and "DO" (that is, use whatever appears on that line as a T/Maker command).

Let's call this line the "FIRST DO LINE" for convenience.

The central section of the file--which appears between the <form> line and the first <end> line--is a FORM DEFINITION. It provides a screen of instructions and, in this case, a single blank field to fill-in the sought-after file's name. In the context of the database command method, let's call this a "COMMAND SCREEN."

The bottom section of the file--which appears between the <record> line and the first <end> line--is a RECORD DEFINITION. This file's record consists of a single line: "get {file.n} edit".

How does the file work? Let's watch it in operation. Suppose you have WORK.1 on your computer's screen and you go into the UPDATE mode. WORK.1's Command Screen appears. It asks you to fill-in the name of the file you seek. In T/Maker's database system, filling-in the filename will in turn create a data record down below WORK.1's record definition. Say you filled-in the filename "LETTER.1". In that case, a new bottom line would appear in the WORK.1 file:

THE FILE WORK.1 AFTER UPDATE:

```
18 do


<record>
get {file.n<    } edit
<end>
get LETTER.1 edit              <----- new bottom line
```

Call this new bottom line a "Command Record."

After filling-in the file name, you would QUIT the UPDATE mode and invoke the DO command. That will cause T/Maker to go to the FIRST DO LINE, making that line into a command. This DO command, in turn, will send T/Maker to the file's 18th line and make that line into a DO command. As it happens, the file's new bottom line--the Command Record--lies on its 18th line. Finally, then, T/Maker will now be instructed by the Command Record to GET the file called LETTER.1 and go into the EDITOR--which achieves our initial objective.

All this may seem like an inordinate amount of work simply to GET and EDIT a file! In fact, however, this approach becomes quite convenient for more complex and time-consuming applications.

A Correspondence System

Suppose, for example, you wanted to construct a T/Maker system that streamlined correspondence writing by automatically preparing a standard letter format with the correct address filled-in. All you would do to use this system is provide three kinds of information: (1) the addressee's name, (2) the new letter's filename, and (3) the date. This information is keyed-in on a Command Screen, and T/Maker does the rest.

As it happens, three separate files are necessary to make the system work. You will need a database file that contains all your address information (herein called DATABASE.1). You will need a file that contains a good, standard format for letters (herein called LETTER.FOR). Finally, you will need the database command file that actually directs the task (herein called LETTER). The LETTER file would look something like this:

FILENAME: LETTER

```
start this line at column 80 -->    delete temp rename temp 49 clip b save 2 do

          THIS IS THE FILE FOR AUTOMATICALLY ADDRESSING AND FORMATTING
                                  A LETTER

                    Type UPDATE to enter addressee's data.

                    Type DO after having entered addressee's data.

leave enough lines here so that the form does not appear on the screen when you first GET the file

<form>

                          TO ADDRESS AND FORMAT A LETTER:

   1.  Please fill-in the first and last name of the addressee below.
       Also, fill-in the name you wish the filename you want the
       to be stored under.  Next, fill-in the date.

                  FIRST NAME:  {firstname                }

                   LAST NAME:  {lastname                        }

              LETTER'S FILENAME:  {file.n<    }

                          DATE:  {Date<           }

   2.  Finally, QUIT the UPDATE mode and DO.
<end>
<record>
 Date = "{Date<           }"
 the next three lines should be placed on one line in your file to be a single DO line
create {file.n<    } insert b:letter.for load temp select b:database.1
when firstname contains {firstname<                } and lastname contains
{lastname<                      } end find <her clip before find : edit
<end>
```

Let's take a closer look at the LETTER file.

The top area of the file -- from its second line to the <form> line -- is what you see when you initially GET the file. This is called the file's "frontispiece." It tells you the file's function, what to do next, and the file's name. Notice that the FIRST DO LINE should be shifted to the right, out of view on an 80-column screen. This reduces clutter.

The second section of the file is the COMMAND SCREEN, running from the <form> line to the first <end>. It tells you what to do and provides places to input information.

The third part of the file is the record definition--running from the <record> line to the final <end> line.

What does the record definition accomplish? Suppose you fill-in the COMMAND SCREEN that you are writing "Frank" "Smith," that the filename for the letter should be "frank.let," and that the date is "August 1, 1985." Once you've filled-in this info, a two-line COMMAND RECORD will appear down below the file's final <end> line. The bottom of the file will now look like this:

```
<record>
Date = "{Date<          }"
create {file.n<    } insert b:letter.for load temp select . . .
<end>
Date = "August 1, 1985    "
create frank.let    insert b:letter.for load temp select . . .
```

Now, quit the UPDATE mode and invoke the DO command. This sends T/Maker to the top of the file, where it finds the first DO line and the following instructions: (1) RENAME the working file to TEMP, (2) go to line 49--which is the 'Date = "August 1, 1985 "' line, (3) clip away everything above this line, (4) SAVE the new two-line file, and (5) go to the new file's second line--a Command Record--and DO.

What does this Command Record accomplish? The first command reads, "create frank.let insert b:letter.for." Thus, the Command Record begins by creating a file named "frank.let." Next, it inserts a file called LETTER.FOR--this being the file that contains a fully prepared letter format in Record Definition form. LETTER.FOR looks like this:

FILENAME: LETTER.FOR

```
<record>
.length 52
.top 2

  {firstname              !} {lastname                          }
  {Date<          }
  Page #
```

(this file continued on next page)

(continued from last page)

```
.end

                                                    {Date<          }

  {firstname              !} {lastname                              }
  {add1                                          } }}
  {add2                                          } }}
  {add3                                          } }}
  {add4                                          } }}

  Dear {firstname!              }:
<end> ..<here>
```

The next bit of the Command Record reads "load temp," which LOADS the date into the FRANK.LET file from the TEMP file.

The next part of the Command Record reads "select database.1 when firstname contains {firstname< } and lastname contains {lastname< } end." This command selects address data from a file called DATABASE.1, where, as mentioned, the user has stored all his address information. Incidentally, using "contains" rather than the "= sign" in this command has a notable advantage. If you happen to forget an addressee's first or last exact name or exact spellings, you can get away with initials.

Finally, the Command Record says "find <her clip before find : edit." This series of commands merely brings the cursor down to a convenient place to start typing the body of the letter. Incidentally, don't write out all of "<here>," including both wedges, in this part of the COMMAND RECORD. A full <here> will inadvertently place a functioning <here> command in your LETTER file, which will confuse T/Maker's SELECT command no end. The "<her" is adequate for the FIND command anyhow.

How Is LETTER Used?

Correspondence is now much easier. All you need to do to format and address a letter is:

1. GET LETTER.

2. Go into the UPDATE mode and fill-in the addressee's name, the desired name of the new file, and the date.

3. Quit the UPDATE mode and type DO.

There are a great many potential applications for the database command method. It is an ideal approach, as we have seen, to problems in which you want merely to enter the particulars of a task but leave the task's actual execution to an already prepared T/Maker routine.

The T/Maker Typewriter

-- Peter G. Dunn

I am an accountant who also works some in programming and consulting. In my work as an accountant, I have to fill out a lot of $%#^@&!* forms. I have been using T/Maker to do some of that work when it is repetitious. However, if I have to fill out a particular form only once a year or so, it is too meticulous to use T/Maker without some assistance -- so out has to come the trusty old typewriter.

Well , the typewriter has been relegated to the junk pile as I have found that combining the attached crude BASIC program and BATCH file with T/Maker gives me an unbeatable answer. This is a GW BASIC program and requires MS DOS.

I say that the BASIC program is crude as it makes no attempt to use the arrow or tab keys. Keep your fingers off them. Put the form in the printer the same way you would with T/Maker and type the command (at the MS DOS prompt) "TYPEIT". You may now use the enter key and space out to your first position on the form. You may use the backspace key. Fill out the form. Press the "~" key. You may now go to T/Maker and GET TYPEIT.FRM. You may correct any errors and then type another copy of your form.

Here are the files:

TYPEIT.BAT

```
BASIC TYPEIT.BAS
(The next two lines should all be on one line but page wasn't big enough)
TMAKER GET TYPEIT.TRF REPLACE '"' '' DELETE TYPEIT.TRF DELETE TYPEIT.FRM
RENAME TYPEIT.FRM SAVE STOP
```

TYPEIT.BAS

```
1 GOTO 25000:'TYPEIT.BAS Copyright @1985 by Peter G. Dunn
7 I$=INKEY$:IF I$="" THEN GOTO 7
8 B$=B$+I$:IF ASC(I$)=13 THEN WRITE #1, B$:B$="":RETURN
9 IF ASC(I$)=8 AND LEN(B$)>1 THEN B$=LEFT$(B$,LEN(B$)-2)
10 IF I$="~" THEN B$=LEFT$(B$,LEN(B$)-1):WRITE #1,B$:CLOSE:SYSTEM ELSE RETURN
100 GOSUB 7:LPRINT I$;:PRINT I$;:GOSUB 200:GOTO 100
(The next two lines should all be on one line but page wasn't big enough)
200 ROW=CSRLIN:COL=POS(0):LOCATE 1,1:PRINT"You may type now, but type ~
to return to menu.";TAB(61);A$:LOCATE ROW,COL:RETURN
25000 DEFINE A-Z:WIDTH LPRINT 132
25010 OPEN "O",1,"TYPEIT.TRF"
25050 CLS:A$="This is TYPEIT.BAS":GOSUB 200:LOCATE 24,1
25100 GOTO 100
```

SALES FORECASTING WITH T/MAKER

-- *Bob Ackerman*

Sales forecasting is an art that requires information from a number of salesmen to be compiled and provided in at least two ways. The company wants to look at future orders in date order. The sales manager needs to appraise each salesman's performance, and have a tool for next month's forecast.

Here's a crack at letting T/Maker do the grunt work. I have not yet made all functions fully automatic, and have not made it menu driven -- yet.

I use the functions of the database and the COMPUTE command, with an imbedded .CLEAN for printing. Figuring out the locations of text, headers, and the records in SAMPLE.RPT was a little tedious, but all ended well. This is not perfect, but for the moment it meets my needs.

Some notes about the system:

1. People speak of months like FEB and JUL, so the record has both "BOOK" and "BMO". UPDATE collects month names from the keyboard and SET converts them to format 8509. Then a date sort is made by SAMPLE.RPT.

2. I will later automate the SET function so that I do not have to remember to do it after each UPDATE. Ideas are solicited.

3. DO lines in SAMPLE.RPT collect data from SAMPLE.DAT, order it by date, insert the file called TOTALS, and file it all away in the file called THISMO.

4. TOTALS places the last line in the report file for COMPUTE to sum.

5. The numbers 5 and 1 in the second DO line find the last line without dropping the reference to the XXXX in the record definition.

6. A bogus record in SAMPLE.DAT permits sorting on the SE or date fields, and still be able to find the end of the file. The rest of the lines help me to figure out what is going on when I am debugging changes to the system.

7. CAT1, 2, and 3 columns are used for different probabilities of business. I use 90, 70, and 50%, but anything applies. I have chosen to pick up only the first comment line in the report.

The following two pages contain the files:

THE FILE SAMPLE.DAT

```
<FORM>
                    FORECASTING DATABASE
                    ========== ========

                   SLS ENGR       :{#SE}:
                   ACCOUNT        :{ACCOUNT                }:
                   PRODUCT        :{PRODUCT     }:
                   BOOK MO        :{B#OOK}:
                   $CAT1          :{>CAT1}:
                   $CAT2          :{>CAT2}:
                   $CAT3          :{>CAT3}:
                   COMMENTS       :{<COMM1                 }:
                                  :{COMM2                  }:
                                  :{COMM3                  }:
<END>
<RULES>
BMO = '8507' WHEN BOOK = 'JUL'
BMO = '8508' WHEN BOOK = 'AUG'
BMO = '8509' WHEN BOOK = 'SEP'
BMO = '8510' WHEN BOOK = 'OCT'
BMO = '8511' WHEN BOOK = 'NOV'
BMO = '8512' WHEN BOOK = 'DEC'
BMO = '8601' WHEN BOOK = 'JAN'
BMO = '8602' WHEN BOOK = 'FEB'
BMO = '8603' WHEN BOOK = 'MAR'
BMO = '8604' WHEN BOOK = 'APR'
BMO = '8605' WHEN BOOK = 'MAY'
BMO = '8606' WHEN BOOK = 'JUN'
<END>
<RECORD>
1 SE             {#SE}
2 ACCOUNT        {ACCOUNT                 }
3 PRODUCT        {PRODUCT     }
4 BOOK MO        {B#OOK}
5 BMO            {BM#O}
6 $CAT1          {>CAT1}
7 $CAT2          {>CAT2}
8 $CAT3          {>CAT3}
9 COMMENT        {<COMM1                  }
10               {COMM2                   }
11               {COMM3                   }
<END>
1 SE             HR
2 ACCOUNT        T/Maker Company
3 PRODUCT        5" diskettes
4 BOOK MO        AUG
5 BMO            8508
```

(this file continued on next page)

```
 6 $CAT1              11.5
 7 $CAT2
 8 $CAT3
 9 COMMENT          DSDD for T/Maker
10
11
 1 SE               XX
 2 ACCOUNT          XXXXX
 3 PRODUCT          XXXXX
 4 BOOK MO          XXX
 5 BMO              9999
 6 $CAT1
 7 $CAT2
 8 $CAT3
 9 COMMENT          bogus record for find-
10                  ing end of file
11
```

THE FILE SAMPLE.RPT

```
                                SAMPLE.RPT
SELECT SAMPLE.DAT END ORDER BMO FIND 9999 DROP 9999 INSERT TOTALS 1 DO
5 DROP XXXX 1 DELETE THISMO RENAME THISMO COMPUTE SAVE 4 PRINT IT
<RECORD>
+       {#SE} {ACCOUNT              }{PRODUCT     } {>#BOOK}{>CAT1}{>CAT2}{>CAT3}
        XXXX                                        {BMO}
<END>
(.INDENT 10 here to line up columns)
SE ACCOUNT                PRODUCT        BOOK    CAT1    CAT2   CAT3    COMMENTS
                                          MO      $K      $K     $K
-- -------------------- ------------ ----    ----    ----   ----    ------------

(.CLEAN here to permit COMPUTE then PRINT)
EX                                                  999.9  999.9 999.9
ZV
<HERE>


                                    TOTALS
=============================================================================
                                                    ----   ----  ----
=                                TOTAL FORECAST
                                                    ====   ====  ====
(.NOCLEAN here)

(end of TOTALS)
```

DRESSING UP THE T/MAKER SCREEN

-- Bert Zitek

While reading up on the requirements for a bulletin board system I came across a brief mention that the ASCII Graphics characters were used to dress up the welcome message screens.

A quick check in one of the manuals and I found that if you hold the 'Alt' key down and enter the ASCII number on the keypad, that the graphic symbol appears on the screen when the 'Alt' key is released. *(Ed. note: this can also be accomplished on some machines by using the high-bit key followed by a 1, 2, 3, 4, or other character. It is easiest simply to experiment to determine which characters produce graphics.)* I didn't get much sleep that night.

But will it work on T/Maker? It sure did -- and I think it is a big improvement when compared to the punctuation characters that I had been using to dress up the screen.

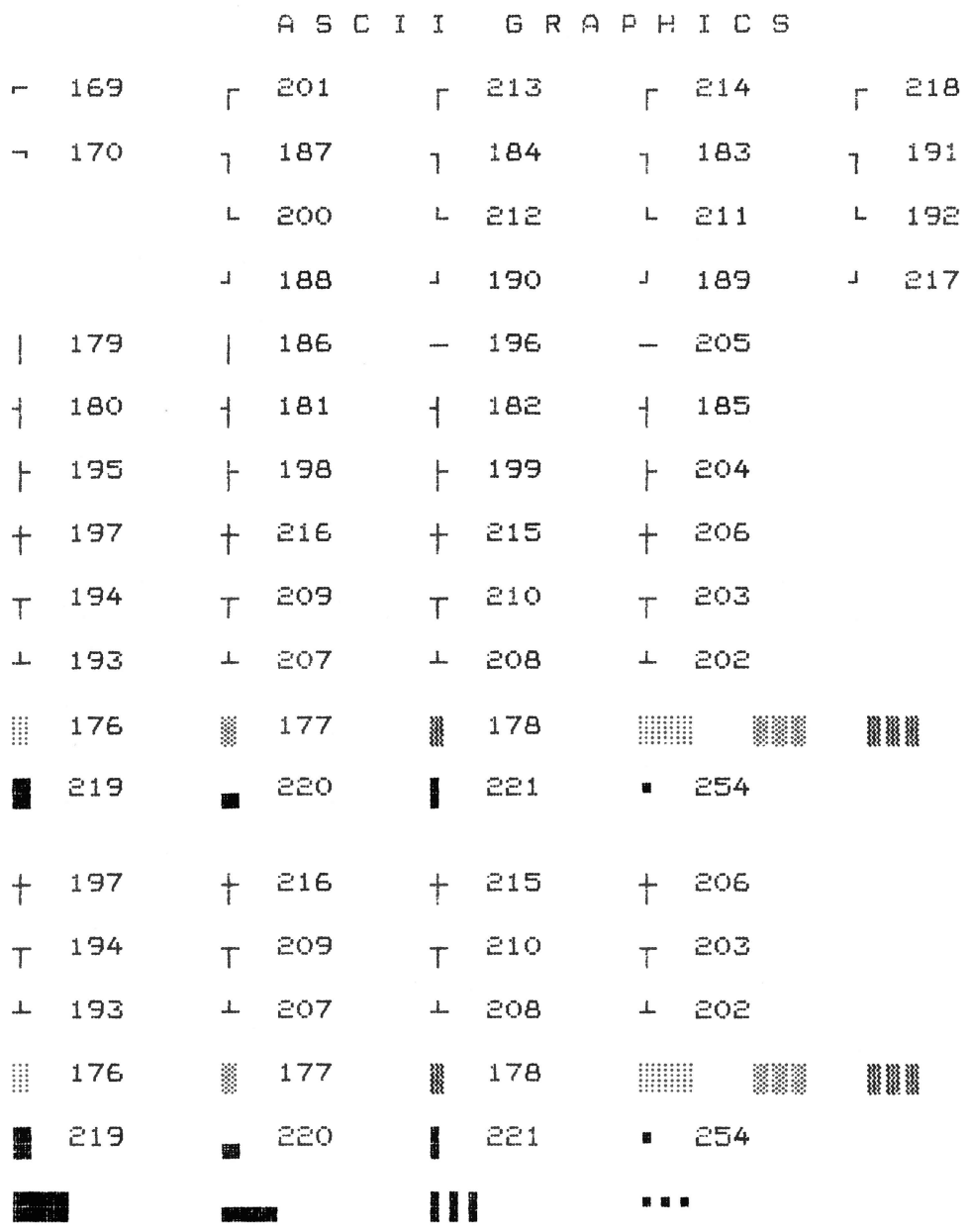
Another plus is the ease of entering all of the ASCII codes using the Macro function provided by T/MAKER.

I cataloged 42 ASCII graphic symbols for boxes and 7 for shading. If you try to print a screen that has been dressed up with ASCII characters you will get various letters of the alaphabet in italic type rather than a reproduction of the screen. I used a public domain program called 'PRTSCFX' which produces almost an exact replica of the screen. I don't usually print out my entry screens, but when I did, those punctuation characters drove the printer nuts trying to follow the print commands. (On some computers, the PRTSCRN key will do the same thing, depending on your printer and computer.)

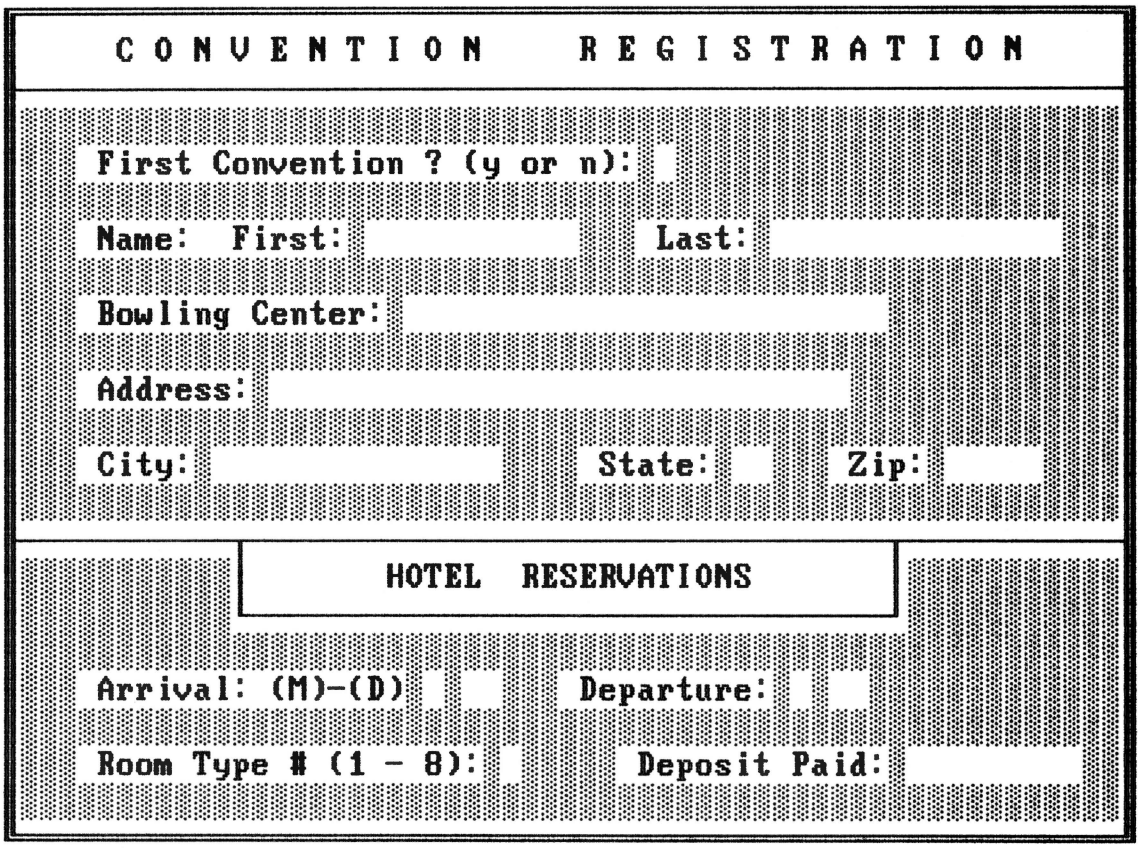
(One quick tip for those who do not want to take the time to use ASCII characters to dress up their screens: Use parentheses around the field brackets or simply put a period at the end of a bracket that requires entry of text. That way you will know before typing over into the next field that the input is too long.)

On the next three pages are the ASCII codes, along with a sample of a screen as it appears un UPDATE mode and as it appears in the file.

ASCII CODES FOR DRESSING UP A SCREEN:

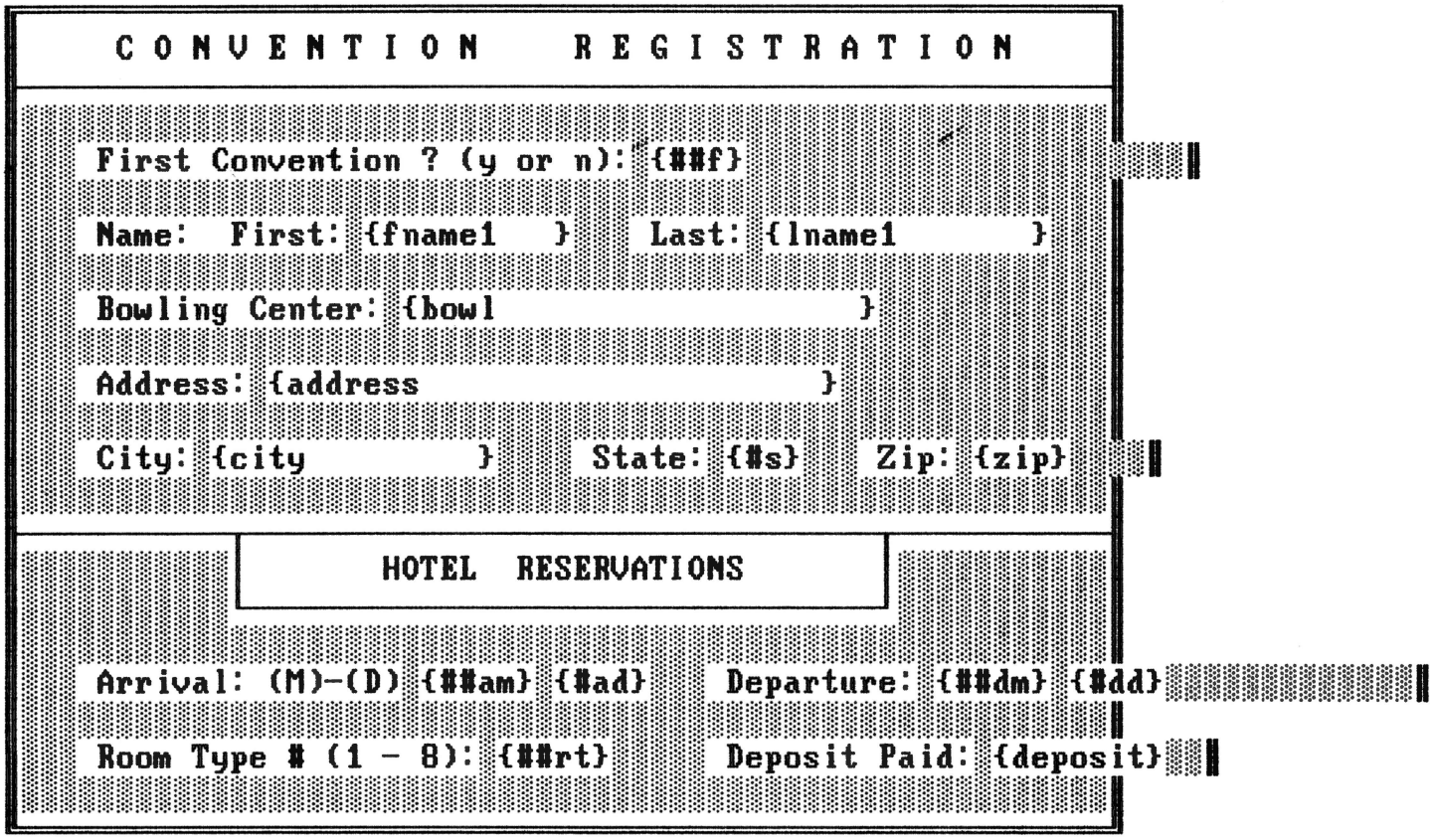


This is an example of a dressed-up screen, as it would appear in the UPDATE mode:



The Form Definition to create prior screen:

```
<form>
```




```
<end>
```

Notice that some of the lines require the characters to extend beyond what appears to be the logical place to put them.. This is because the use of the # mark in the field to shorten it makes the rest of the characters in the field effectively disappear for spacing purposes.

Those of you with version 4.03 and an IBM PC can see the use of this technique in some of the MODELS files.

T/Maker Consultants

T/Maker consultants are T/Maker specialists who are willing to consult on your applications for a fee. Please contact them directly for any further information on their work. If you are interested in being listed as a consultant, please send us your applications (on paper and on disk) and a brief statement about your area of specialization.

Don Baack, 6495 S.W. Burlingame, Portland, OR 97201, (503) 244-2741, TELEX 277331 SMA UR: *Manufacturing; operations analysis; cost accounting; inventory control; timber and wood products.*

Dick Danielson, Strategic Advisory Consultants, Inc., P.O. Box 137, Star Prairie, WI 54026, (715) 248-3434: *Fully integrated accounting system including general ledger, financial statements, accounts receivable, account analysis, operations analysis. Multiple divisions, accounts, and ledgers can be accommodated.*

Gus Korman, English Only Computer Co., 5222-1 Lindley Ave., Encino, CA 91316, (818) 344-2422: *Accounting system including many of the most popular accounting modules.*

D. Michaut, Dept. Org. Societe Generale Des Coop., 27, 33, quai A. le Gallo, 92517 Boulogne-Billancourt Cedex, FRANCE (1) 604-9178: *Retail and warehouses, Price and margin management, stock management, merchandising, invoicing, mailings.*

Robert Payne, BCS, 1210 Smith Street, Charleston, WV 25301, (304) 343-9471: *Lawyer time accounting, general office accounting, A/P, A/R, payroll, general ledger.*

Ron Roizen, 1818 Hearst Street, Berkeley, CA 94703, (415) 848-9098: *Automating small businesses, including accounts receivable/billing, mail-order accounting and inventory. Survey research, tracking organization membership, dues. Also training and documentation.*

Elyse Sommer, Box E, 962 Allen Lane, Woodmere L.I. NY, (516) 295-0046: *Text-related data base use; custom macros with keyboard enhancers; on-screen writing "helpers" for correspondence, reports, manuals; fact sheet outlines; phrase files; boiler plate.*

Emil Widmer, CFM, Baarerstrasse 45, Postfach 708, CH-6301 Zug, Switzerland, tel:042 21 08 87: *consulting on management of uses of computers, management seminars using T/Maker in Switzerland and Germany, engineering and design, production engineering, personal resources, finance.*

Robert D. Williamson, Can-American Electronics, 7220 Taft Street, Hollywood, FL 33024, (305) 966-5588: *Small business and manufacturing software. Hardware.*

T/Mug Back Issues

If you found this issue useful, you might want to consider picking up the back issues. T/Mug has been published since 1982, each issue aiming for a mix of articles for the novice T/Maker user through the expert.

The T/Mug also serves as a forum for users of particular machines and in particular industries. We try to include the applications most requested in letters and phone calls. Past issues have included: invoices, checking account management, statistics, personalized form letters, transferring data files, inventory, tracking athletic events, time calculations, and many more. The applications, like the product, have evolved over the years, so you may find the more recent issues to be of more value.

To order, please enclose this form with your check (drawn in US dollars) and mail it to the address below. Customers outside continental US please add $5 per box checked for shipping.

Back Issues -- Please send me:

[] Back issues: 1984 - present (six + issues) $15.00

[] Back issues: 1983 (five issues) $10.00

[] Back issues: 1982 (five issues) $10.00

T/Mug Subscriptions:

You might also want to subscribe (or renew your subscription) to the T/MUG newsletter. This entitles you to six bimonthly issues like the one in your hands.

To order, please enclose this form with your check (drawn in US dollars) and mail it to the address below. Customers outside continental US please add $20 for first class mailing charges.

Subscription -- please:

[] Enter my subscription for one year $15.00

[] Renew my subscription $15.00 (my T/Mug mailing label is enclosed)

Name_____

Company_____

Street_____

City:_____State_____Zip_____

T/Maker Serial Number(s)_____

Please make check out to T/Maker Users' Group and Mail to

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

T|MUG

T/MAKER USER'S GROUP
T/MAKER COMPANY
2115 LANDINGS DRIVE
MOUNTAIN VIEW, CA 94043

BULK RATE
U.S. POSTAGE
PAID
PALO ALTO, CA
PERMIT NO. 96