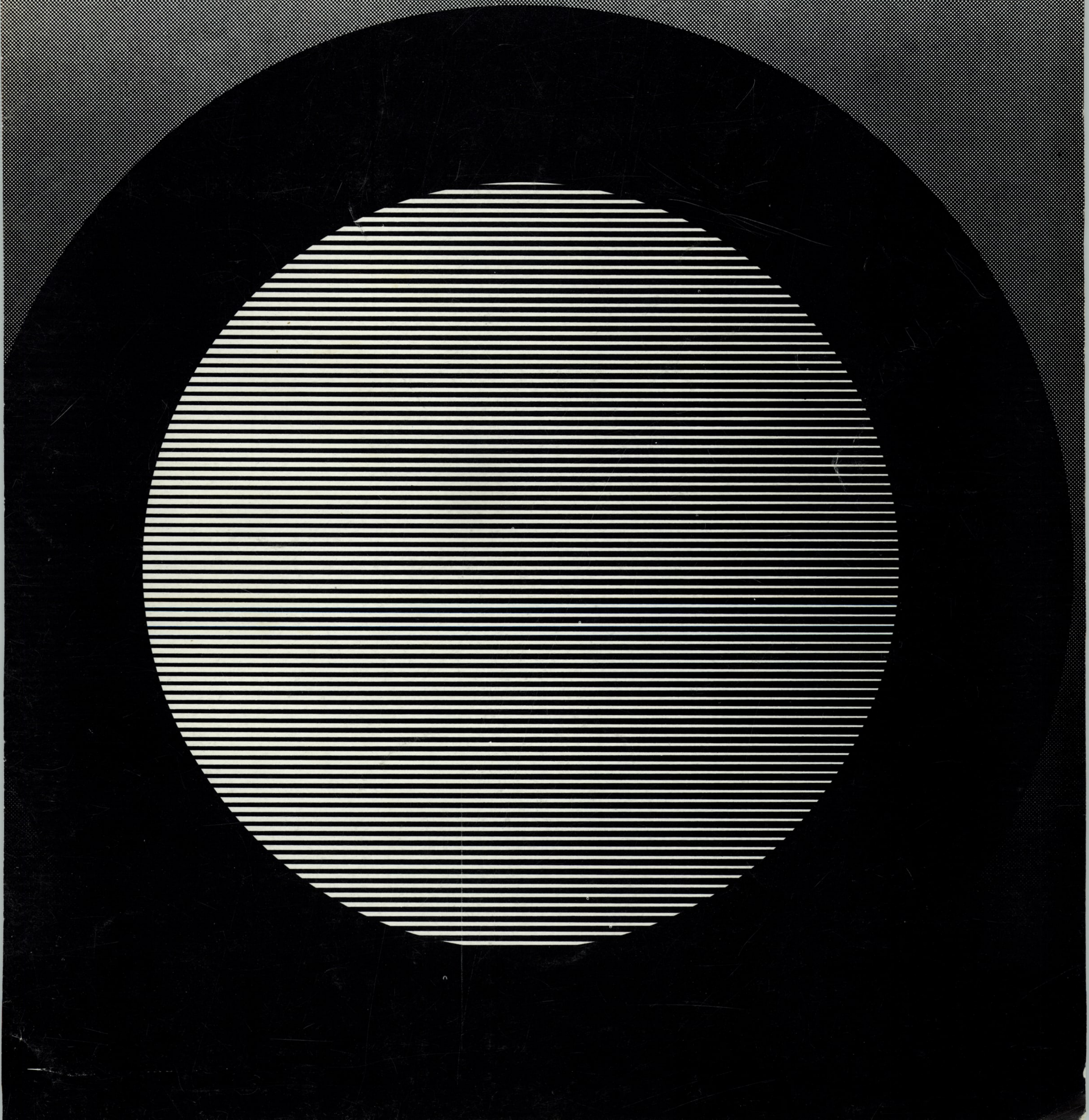
Lifelines®

The Software Magazine™

$3.00 October 1983 Volume IV, No. 5 (ISSN 02479-2575, USPS 597-830)



dBASE II™ made easy!

QUICKCODE™

The dBASE II Program Generator

Now dBASE II is made easy with Quickcode by Fox & Geller. QUICKCODE is a program generator, a computer program which writes computer programs.

FAST AND SIMPLE

With QUICKCODE you can generate a customer database in 5 minutes. Its that fast. All you have to do is draw your data entry form on the screen. It's that simple!

NO PROGRAMMING REQUIRED

QUICKCODE writes concise programs to set up and maintain any type of database. And the wide range of programs cover everything from printing mailing labels and form letters, to programs that let you select records based on your own requirements. There are even four new data types that are not available with dBASE II alone.

YOUR CONTROL

And since you work directly with your information at your own speed and your own style, you maintain complete control. Telling your computer what to do has never been so easy.

QUICKCODE, by Fox & Geller. Absolutely the most powerful program generator you've ever seen. Definitely the easiest to use.

Ask your dealer for more information on QUICKCODE and all the other exciting new products from Fox & Geller.

FOX&GELLER

Fox & Geller, Inc. Dept. LIF 001 ***604 Market Street Elmwood Park, N.J. 07407 (201) 794-8883***

Lifelines The Software Magazine

October 1983 Volume IV, Number 5

Publisher: Edward H. Currie
Editor in Chief: Susan E. Sawyer
Production Manager: Kate Gartner
Technical Editor: Al Bloch
Art Director: Kate Gartner
Typographer: Rosalee Feibish
Cover: Kate Gartner
Dealer/Customer Service Manager: William F. Lampe
Circulation Manager: Trina McDonald
Advertising Manager: William F. Lampe
New Versions Editor: Lee Ramos
Production Assistant: Marcy Rauch
Printing Consultant: Sid Robkoff/E&S Graphics

Lifelines (ISSN 0279-2575,USPS 597-830) is published monthly at a subscription price of $24 for twelve issues, when destined for the U.S. Canada, or Mexico, $50 when destined for any other country. Second-class postage at New York, New York; additional entry paid at Smithtown, New York. POSTMASTER, please send changes of address to Lifelines Publishing Corporation, 1651 Third Avenue, New York, NY 10028.

Editorial

by Edward H. Currie

Electronic Spreadsheets... Solutions In Search of Problems

Art vs. Matter

The microcomputer industry has historically used the end user as the beta-test site for all new technology. As a result the unsuspecting end user has assumed that software and hardware products available from the local computer store and widely discussed in the trade press are safe, reliable and worthwhile. Actually, the new ideas which seem to flow continuously from the microcomputer world are in many cases anything but proven.

Devices such as the mouse require training and some skill and often prove to be less than exciting. Couple this with the fact that little software was designed with such a device in mind and that most operating systems don't assume the existence of such a gimmick and you begin to appreciate the problem.

That's not to say that there isn't a place for the mouse in the repertoire of the microcomputer user; on the contrary, rather, this kind of technology is a solution looking for a problem or problems.

Similarly, color graphics came on the scene with little to recommend it to those who did not have a color and/or graphics printer with which to make hardcopy. It did, however, appeal to that primal fascination that every human has with cartoons. And what about the frequent references to high-res graphics? High resolution graphics does not exist in any widespread sense for micros except for the financially well off, and the term "high" in this case is relative to primitive graphics (i.e. alpha numeric character graphics). High resolution graphics will be upon us when 1024x1024, 2048x2048, etc. screen graphics are widely available, not before. Is there any rational human being who honestly believes that 240x 200 graphics are good for anything other than games? And the list goes on and on....

Imagine if you will that you are hard at work on a sales forecast for the coming fiscal year and have begun with ledger paper and calculator to prepare a spreadsheet. A friend offers to help you by placing a sheet of paper over your worksheet with a rectangular hole in it and informs you that if you need to see a portion of the spreadsheet top just give him the x and y coordinates of the area of interest and he'll, quick like a bunny, move the opening and let you view the area in question. He acknowledges that this is something of a handicap but he'll make up for this by doing all the calculations for you.

You reluctantly agree to accept his help only to discover that each time you institute the most minor changes requiring a recalculation, he responds by recalculating the entire spreadsheet. You are left to contemplate your navel in the interim. Furthermore, because he insists upon calculating the various rows and columns in a specific order the results conflict with your hand calculations due to the way in which you have laid out the speadsheet.

He returns the next day and informs you that you no longer have to give coordinates, just point with your finger. It's here that you discover that the tip of your finger is of finite extent and your ability to aim this device accurately is somewhat lacking given the resolution required to designate specific areas of the document in question.

You resort to comments such "no, a little farther over to the left, and down...no, no, not that far."

Since you have limited visibility of the spreadsheet at all times you suggest to your friend that you can't possibly remember what the 103rd row and 210th column represents, at which point he responds with lightning speed by placing a legend on the left hand side of the window. Thus you now have a reference for identifying cell contents but have simultaneously lost even more of your view of the spreadsheet...and on and on it goes. The more you are offered and accept the more you give up to receive it. This scenario is, of course exactly what happens with computerized spreadsheets.

But hang on, sports fans, it gets even more ludicrous.... Imagine sitting down at your favorite micro and discovering that your recent spreadsheet now has some exciting new features! Through the wonders of modern technology you can now split the screen so that you can look at a bar graph, review a related document and work on your spreadsheet simultaneously. Remember, this spreadsheet is the same one that you couldn't see enough of to begin with....

This is really handy because you happen to be the type who is interested in looking at spreadsheets for P&L, Cash Flow, Departmental Expenses, Advertising, and Operating Expenses simultaneously and associated bar graphics for each while reserving a window for the *New York Times* (after all one must keep up with the *Times*). No problem...!!! Just break the old screen-o-reeno up into fifteen different windows and you can look at the contents of any two rows and columns of each and at the same time at that. What's that you say? How do you know what the rows and columns in any particular window represent and where you are looking on the respective spreadsheets??? Oh come now, what's become of your memory, my boy? Do you expect the microcomputer to do all the work?

By now you have gotten the point. There are obviously a number of issues involved but the key point is that it is the poor end user who is left to find a use for such technology and it is he who establishes whether or not this stuff in the final analysis has any significant and lasting value.

We should all take heed and recognize that this kind of product will not be tolerated much longer by the end-user community. We are coming to the end of a free ride which is based upon the general public's fascination with computer technology. Either the software and hardware presented to the masses will in fact

(continued on page 32)

Feature

PL/I From The Top Down

By Bruce H. Hunter

This is the fourth in a series of articles about PL/I Subset G. This series is rewritten from my book entitled "PL/I From The Top Down," a casual approach to a serious, formal language. For further, more intense study, there are several books available on the full set of PL/I, the most comprehensive one being "PL/I Structured Programming," by Joan K. Hughes (John Wiley & Sons, 1979, Second Edition).

CHAPTER THREE — EXCEPTION PROCESSING OR "BULLETPROOFING" YOUR CODE

Short of equipment failure, there is nothing more exasperating than having a program or utility fall on its face in mid-execution. Your program abruptly ends and suddenly you're confronted with the operating system prompt. Abnormal termination of the program is considered to be the mark of poor programming. How to avoid it is the subject of this article and chapter.

The ultimate goal of a programmer is writing a program that keeps running and stays "on its feet" at all times. A program written in a professional manner can never be allowed to terminate abnormally because valuable data can be lost that way. To prevent abnormal termination, exception processing techniques are used, sometimes referred to as "bulletproofing" the code. Exception processing is a systematic, programmatic approach to ensure that any possible errors are anticipated before they happen. Battle plans are drawn up to figure what you are going to do with each error when it is detected. When an error occurs, the program remains in control. For example, the program can be carefully protected from the input of data or data types that conflict with the internal representation of the data the program expects to see. The program can be safeguarded from data that is "out of bounds" or inappropriate to the program. There are many exception processing techniques.

In other programming languages it's often a chore to perform these various bulletproofing tasks to handle any potential errors. Standard Pascal has no built-in provisions for error handling. C has no error handling provisions per se, and neither does FORTRAN. Most versions of BASIC have one "on error" function which helps a little. Some of the enhanced versions of Pascal and BASIC are a little better. For example, MBASIC has some relatively sophisticated error handling capabilities, although they don't touch PL/I's. Most of the time the programmer is forced to create elaborate little routines to accomplish his exception processing tasks.

PL/I, on the other hand, makes exception processing much easier. It has an entire library of built-in error routines which you can use simply by typing the keyword "on" and the specific error for which you wish to test. For instance, how many times have you written page break routines to test for the end of the page? It's just one more little chore to write the darned thing. In PL/I all it takes is just one line of code, the program statement "on endpage," and this automatically tests for the end of the page. You can test for the end of the file just as easily. Errors such as data overflow or underflow, data type conflict, stack overflow, zerodivide (divison by zero), and undefined file are also easily tested for in your code with the use of the keyword "on" and the specific potential error. If the error is encountered and detected by the program before it can inflict irreparable damage, new data can be requested and the program is able to go on about its business with no loss of data and little loss of time. Errors that can be anticipated and prevented are called "recoverable errors." Some errors are fatal to the program, but that doesn't mean you need to lose program control. For instance, if free memory space is exhausted by the program, the error is fatal, but the programmer does not have to let the program 'crash' with the ensuing loss of data. The program can be kept on its feet long enough to print an appropriate error message, close the files and THEN terminate quietly. Bulletproofing the code with the aid of a good language can be accomplished with little pain. It does not come free, however. A bulletproofed program usually runs at least 20% larger than an unprotected program because all boundary conditions must be anticipated. Let's explore this further by looking at some code. Consider the inherent problems involved in testing input for out-of-bounds conditions. Here's a typical "menu" form often used in programming where input from the operator is required:

```
redo:
  dcl choice fixed;
  put skip edit (
    '1 update',
    ',
    '2 delete',
    ',
    '3 display',
    ',
    '4 return to system',
    ',
    'input choice')
    (9(skip,a));
  get list (choice);
  /*
  */
  if choice <1 | choice >4 then
  goto redo:
```

►

Here the console operator is asked to input the number of his choice (any number from one to four). The input is checked to see if it falls within the bounds set by the program by the "if choice <1 | choice > 4 then goto redo:" statement. At first glance it seems that this code is bulletproofed well enough. But what happens if an alpha character is entered? It wouldn't be the first time "l" was input for "1." If alpha data were entered, PL/I would go into death throes, its last words being "Error 1 Conversion," whereupon your program would roll over and die leaving you with nothing but a system prompt for company. If, at the same time, a file were opened for input or update, the abnormal termination would leave the directory unwritten. Not only would all the data in the program buffers be lost, but so would a good piece of the disk data base. How can you avoid calamities like this? Here is some code which represents one way to combat any possible alpha entry errors in our menu input routine:

```
on error(1) /*data conversion error */
  begin;
  put skip list ('numeric input required');
  goto redo;
  end;
```

The input routine is more completely bulletproofed if this is included. The bounds are protected, and the program can recover if alpha data is input. The "on error()" condition prepares the system for whatever disaster is enumerated as its argument. There is a shorter method of bulletproofing the last routine, however, and I would be remiss not to point it out:

```
dcl
  choice char (1);
redo:
  put skip edit (
   '1 update',
   '',
   '2 delete',
   '',
   '3 display',
   '',
   '4 return to system',
   '',
   'input choice')
   (9(skip,a));
  get list (choice);
  /*
  */
  if choice <'1' | choice > '4' then
    goto redo:
```

The addition of single quotes around the bounds '1' and '4' makes them character. The variable "choice" is changed from fixed to character, and the error-proofing is complete. All keyboard input is character unless PL/I is looking for numeric, so anything is acceptable to the input statement "get list." If the ASCII values fall above or below the one to four range, the input causes the program control to go back to redo, which is a backwards goto. In the late 60's and into the mid 70's, a great deal of pioneering work and reeducation was done in the field of structured programming by such notables as Richardson, Butler and Tomlinson. The backwards "goto" fell under heavy attack as the cause of difficult-to-understand program code because of its ability to "spaghetti string" the code into a hopeless criss-crossing of transfer of control. This condition not only makes the program impossible to understand and decipher when changes are made to the program, eventually one of the "spaghetti strings" is going to be cut inadvertently. Then, as Hogan would say, the program falls down and dies. Disciples of the new structured programming preached the gospel of "gotoless" programming which became synonymous for structured programming.

Structured programming is not necessarily "gotoless," however. In PL/I there is no BREAK command as there is in C to transfer control out of the loop or procedure. A call to a separate procedure avoids a "goto," but sometimes it can be a "copout" for a "goto," and often it is just as confusing. Let's look at another way to avoid a backwards "goto" and still solve our input testing problem. A program segment within a loop (it's placed within a loop for the purposes of avoiding a "goto") also works, but it doesn't always clarify the code as the following pseudocode segment shows:

```
do while (bad__data)
  write " input a number from 1 to 6 "
  read number
  if number > 0 .and. number < 7
    bad__data = FALSE
end__do
```

The loop provides a trap to hold the program into constantly asking for good data and not continuing until it gets it. Here's a "gotoless" solution, but how good is it? It works fine, but at first glance it looks as if it were written to perform a task repeatedly. It's not immediately recognizable as an input routine to someone reading through the code, so this section will probably have to be reread to determine what the programmer wanted to accomplish. I like code to be as simple as possible. I hate wasting my time puzzling over someone else's code. While sophisticated and clever programming solutions are gratifying for the person writing the code to discover and use, if anyone else is going to have to read that code, throw out the clever stuff and make it simple and clean. Ninety-nine percent of the time, it'll work just as well. Let's look at a solution that has a short backwards "goto," short meaning that it is very close to its label:

```
redo:
  write "input a number from 1 to 6:
  read number
  if number < 1 .or. number > 6
    goto redo
```

The second solution is immediately recognizable as an input routine. The bounds are clear, and it is immediately apparent what is happening here — if the number is less than one or greater than six, go back and get it again. I'm going into this in detail because the best way to do PL/I exception processing is with the use of the backwards "goto." We live in an imperfect world, receive imperfect data, and sometimes the best way to deal with it is in an "imperfect" way.

Let's examine the list of errors PL/I lets you anticipate and capture in your code. After your program is compiled, you are faced with two categories of run-time errors: the ones that are recoverable and the ones that are not. The nonrecoverable errors are deadly — they'll stop the execution of your program cold, and the only option you have is rewriting the program and recompiling it. The recoverable errors are the ones which can be intercepted by using what the PL/I-80 manual calls the "ON unit." You've seen the "ON unit" in action already in an early version of our menu, where "on error(1)" was shown to be one effective bulletproofing method for screening input data. Using the keyword "on" with the error you want to anticipate and avoid is one of the powers PL/I puts at your coding fingertips. Here are a few of the error conditions which can kill a program with no exception processing techniques:

ERROR (1)
DATA CONVERSION
The number one slayer of programs.
Data input or processing has uncovered data types that do not conform to each other.
Character data entered where numeric is expected is a typical conversion error.

ERROR (2)
I/O STACK OVERFLOW
There is not much of a cure for this one except once it has occurred, increase the stack height through the procedure options statement, proc options (stack(1024));
The default stack is 512 bytes.

ERROR(3)
The argument of a transcendental function has been found to be out of range.
Look for such gremlins as a negative value passed to a square root function.

ERROR(4)
I/O CONFLICT
The number two slayer of programs if you do a lot of file work. The attributes (descriptors) of the file in the file open statement do not match the put/get or read/write statements. This one goes away when you understand files better.

ERROR(5)
FORMAT OVERFLOW
In spite of the fact that this is listed as a "recoverable error" (your program will not stop execution, if an ERROR(5) is encountered), "format overflow" due to excessive nesting of embedded formats cannot be saved with error recovery in the code. You're going to have to simplify your program and recompile it.

ERROR(14)
UNSUCCESSFUL WRITE
This one tells you if it can't enter the data to disk for whatever reason.

ERROR(15)
FILE NOT OPEN
The usefulness of this one is obvious.

ENDFILE
This one is handy not only to keep your program from crashing but to cause an action to happen when the specific condition is encountered such as simply exiting a loop when the physical or logical end-of-file is encountered.

We took a short look at endpage, and other errors are undefinedfile, fixedoverflow, several types of zerodivide errors, two types of overflow errors, and two types of underflow errors. This all sounds pretty exciting, but you can't get too carried away with using the "ON unit." If more than 16 "on" conditions exist, you will get another error message:

CONDITION STACK OVERFLOW

There is no "on" condition for that one! But there is a switch which turns "on" conditions off called "revert." Let's look at a code fragments showing "revert" and showing some specific uses of some of these "on" conditions:

```
  on endfile (db.dat)
    goto count__records;
i = 1;
  do while (TRUE);
    read file (bd.dat) into (dummy) key to i;
    i = i + 1;
    end: /* do */
  count__records:
  number__of__records = i;
  revert endfile (db.dat);
```

This is a file record counter. Note that the endfile used in the program segment above gets you neatly out of the loop and on your way when the file is read. Let's look at the endpage and endfile conditions.

```
  on endpage (printer)
    begin;
    put file (printer) list ('↑I↑I↑I page ',page__no,'↑L');
    page__no = page__no + 1;
    end;
  on endfile (data)
    goto end__loop:
  do i = 1 to 32467;
    read file (data) into (data__structure) key from (i);
    put skip list (inventory__number, descript,
      amount);
    end; /* do while */
  end__loop:
  revert endfile(data);
  revert endpage;
```

The "on endpage" causes the program to print the page number and then do a formfeed (↑L) at the end of the page. The "on endfile (data)" takes control down to where the revert statements shut off their own on conditions. Let's look at undefinedfile. Using "on undefinedfile" is a sure way to guard the program against having the name of a nonexistent file input:

```
 dcl
    filename file variable;
 on undefinedfile (filename)
    begin;
    put skip list
      ('filename is not a valid filename');
    goto reinput;
    end;
 reinput:
 put list ('enter filename: ');
```

Zerodivide, fixedoverflow, overflow, and underflow are errors you get when performing mathematical operations. Zerodivide refers to "division by zero," and there are three zerodivide errors corresponding to the numeric ►

data types. Here's a whimsical code fragment testing for a zerodivide error:

```
on zerodivide
   goto next;
do while (TRUE);
   read file (stat.dat) into (stat__structure) keyfrom (i);
   persons\car = nbr__people / cars;
   put skip list ('people per car', persons\car);
   next:
   i = 1 + 1;
   end; /*do*/
```

This fictitious statistics calculator would get a division by zero if the household had no car, but the "on zerodivide goto next;" statement simply skips around the troublesome zero entry and pretends there are no households without cars. (Once I had a boss who used to think like that.)

Underflow and overflow are a lot more complex in handling and require an in-depth understanding of data types and the effect of arithmetic operators on them. For those with PL/I-80, Chapter 12 of Digital's "Applications Guide" on decimal computations will be of help. Arithmetic errors such as division by zero, underflow and overflow are usually terminal, however, and are best avoided by rejecting out-of-bounds data at the start. Take great care in insuring that data will be rejected on input if it is too small or too large to calculate so you don't flirt with the danger of overflow and underflow:

```
put skip list ('input number :');
get list (number);
   if number > 100000000 | number < .00000001 then
      call re__enter;
```

The above example guarantees that the input will be within some sort of bounds. Again, this is critical to assure that the input is sensible and that the program avoids overflow and underflow problems. Here's another example of playing it safe this way:

```
redo:
put skip list ('input month as number :');
get list (month);
if month > 12 then
   goto redo;
```

Common sense error proofing

All the simple to fancy exception processing techniques aside, we can never forget about simply "using our heads" to error proof programs. You don't always have to "refer to the manual to look up the appropriate recoverable error..." and so forth. Common sense rules. What if you want a simple yes or no input? Look at the following code fragment and see if you think it will work:

```
put skip list ('continue yes or no: ');
get list (answer);
 if answer = 'yes' then
   call what__ever;
   else
     if answer = 'no' then
       goto however;
```

Want to bet a box of floppies it won't make it through an hour's worth of inputting without falling through to the next line? It can't handle uppercase, abbreviated, or inappropriate input.

```
dcl
   reply char(3) var;

redo:
put skip list (. . . . . . . . . . . . . . yes or no :');
get list (reply);
if substr (reply,1,1) = 'y' | substr (reply,1,1) = 'Y' then
   call what__ever;
   else
     if substr (reply,1,1) = 'n' |
        substr (reply,1,1) = 'N' then
        goto who__ever;
        else
        goto redo;
```

Granted this is a lot of coding to get a simple yes or no answer, but sometimes that's what it takes to get the job done well. A lot of unnecessary coding can be avoided by numeric input as character for branching such as:

```
dcl
   reply char (1);

.
.
.
retry:
put skip list ('1 to continue, 0 to make corrections :');
get list (reply);
if reply = '1' then
   call next__proc;
   else
  if reply = '0' then
    call reenter;
    else
    goto retry;
```

This approach avoids conversion errors for non-numeric input, string manipulation, and upper/lowercase checking or conversion.

One fact to always bear in mind is this. To take a program from a simple functioning algorithm to a bulletproofed, idiot-proofed, exception processed piece of commercial quality program can easily double the size of its input procedures.

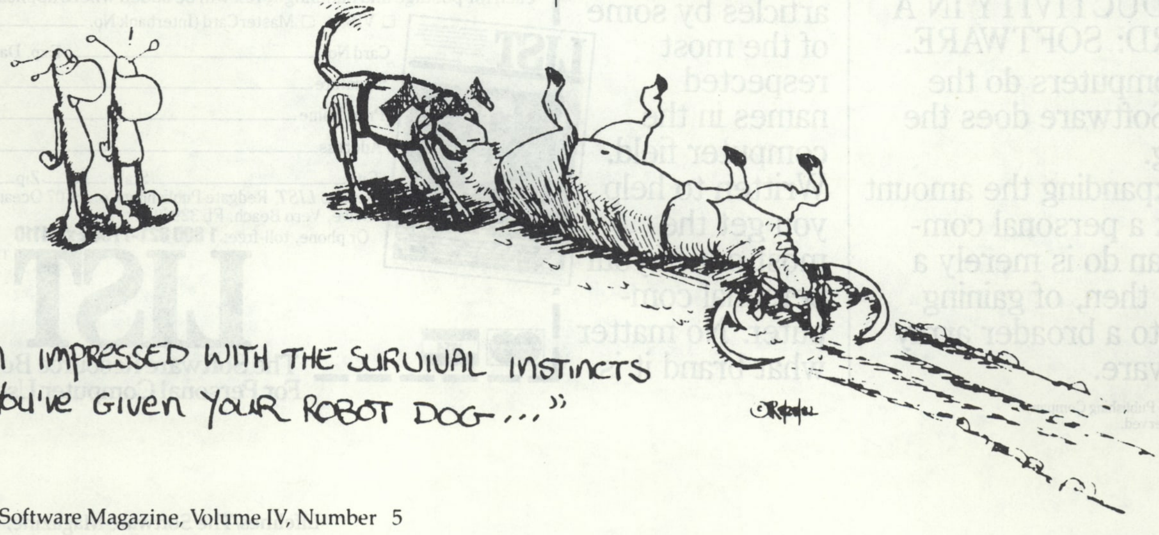
As we discussed before, there is a problem beyond getting the program to function without falling down, and getting data within reasonable bounds. That is the verification of data on entry. We all know the GIGO rule of "garbage in, garbage out." Don't forget to give the console operator the opportunity to read the data entered and change it if need be before allowing the program to process it further. An input procedure, followed by a verification procedure, followed by a re-entry procedure, will do the job, but with a little planning the input parameter can do double duty:

```
mail__list:
  proc options (main);
  dcl
  1 data__structure,
    3 name char (64) var,
    3 address char (48) var,
    3 phone char (16) var;
% replace
  CLEAR by '↑L',
  TRUE by '1'b;
dcl
  disk__file file,
  i fixed;
put list (CLEAR);
put skip list ('mail list data entry');
open file (disk__file) direct output env (f(128));
do i = 1 to 32767;
  call input;
  call verify;
  end; /*do*/

input:
  proc();
  put list ('name: ');
  get edit (name) (a);
  if name = EOF then
    call close-file;
  put list ('address: ');
  get edit (address);
  put list ('phone: ');
  get edit (phone)(a);
  end input;

verify:
proc();
  dcl
    reply char(1); /*local variable*/

  put list (CLEAR);
  retry:
  put skip (5) edit (
    'Name ..........',name,
    '',
    'Address...........',address',
    '',
    'Phone ..........',phone',
    '',
    'enter <1> to continue, <0> to correct')
    (7(skip, a));
  get list (reply);
  if reply = '1' then
  call disk__enter;
  else
  if reply = '0' then
        call input;
      else
        goto retry;
    end verify;

  disk__enter:
    proc();
    write file (disk__file) from (data__structure)
      key to (i);
    end disk__enter;
  close__file:
    proc();
    close file (disk__file);
    stop;
    end close__file;

  end mail__list;
```

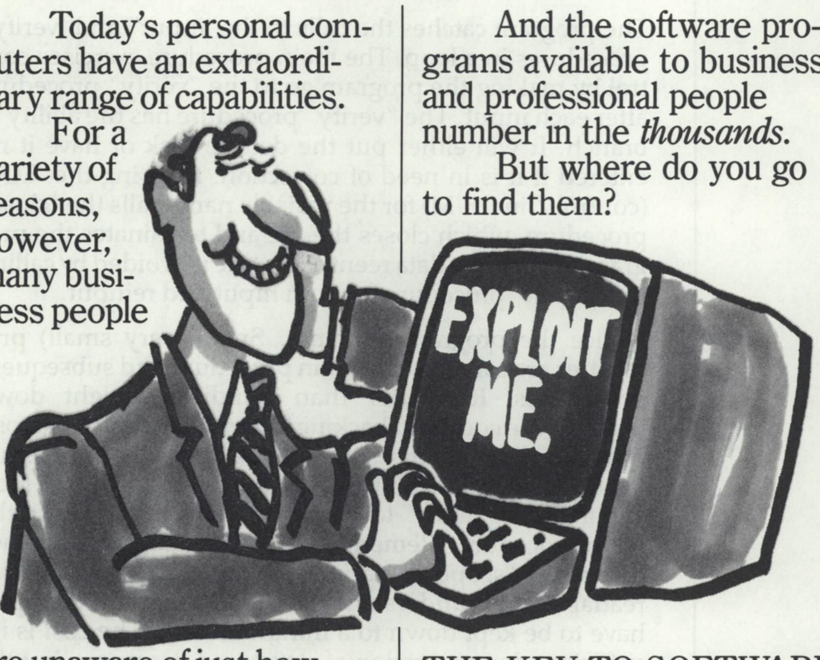
The program catches the calls to the 'input" and 'verify" procedures in a loop. The main procedure exercises control by making the program go to the "verify" procedure after each input. The "verify" procedure has the ability to branch. It will either put the data to disk or have it re-entered if it is in need of correction. Entering the 'EOF' (control Z in CP/M) for the variable name calls the "close" procedure, which closes the file and terminates the program. A separate data reentry routine is avoided by calling the "input" procedure for both input and reinput.

Notice the program structure. Small (very small) procedures are called by the main procedure and subsequent procedures. It's better than grinding straight down through the code and backing up with backward "gotos." This not only avoids "square code" but places program control primarily with the main procedure. It allows local variables like "reply" to keep their value hidden from the main procedure. (Remember the installment in which we discussed "scope of variables.") To keep program size readable and understandable, the program examples have to be kept down to a minimum size. The cost is insufficient demonstration of structured programs, but the goal of highly structured programs should never be forgotten!

Bulletproofing, whether by exception processing, or good sense programming, is a necessity for high grade programs. It is certainly a lot easier to do it with a language that makes it convenient for you. In the next article, we will look at edited I/O, plain, fancy and picture. ■


"I'M IMPRESSED WITH THE SURVIVAL INSTINCTS YOU'VE GIVEN YOUR ROBOT DOG..."

YOU SPENT $4,000 ON A PERSONAL COMPUTER. FOR ANOTHER $12.50, YOU CAN GET YOUR MONEY'S WORTH.



Today's personal computers have an extraordinary range of capabilities.

For a variety of reasons, however, many business people are unaware of just how much their computers are capable of.

As a result, they aren't realizing the full potential of their investment.

THE KEY TO GREATER PRODUCTIVITY IN A WORD: SOFTWARE.

Computers do the work. Software does the thinking.

Expanding the amount of work a personal computer can do is merely a matter, then, of gaining access to a broader array of software.

And the software programs available to business and professional people number in the *thousands.*

But where do you go to find them?

THE KEY TO SOFTWARE IN A WORD: *LIST.*

LIST is the first publication that puts software first.

It contains articles by some of the most respected names in the computer field. Written to help you get the most out of your personal computer. No matter what brand it is.

No matter what you need it to do.

More importantly, *LIST* contains the *LIST Software Locator,*™ a comprehensive guide to over 3,000 personal computer programs—conveniently indexed by application, industry, operating system and hardware. You'll find detailed descriptions of applications software that pertains specifically to the type of business you're in. And the type of needs you have.

LIST is sold at leading computer stores and bookstores. Or, you can phone our toll-free number (1-800-821-7700, Ext. 1110) or send in the coupon below, and receive a copy by mail. The price, exclusive of postage and handling, is $12.50.

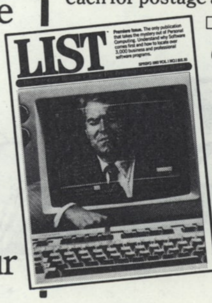
Which, when you think about it, is a pretty small price to pay for something that can maximize a much larger investment.

LIST *is published by Redgate Publishing Company, an affiliate of E.F. Hutton.*

I'D LIKE TO GET THE MOST OUT OF MY PERSONAL COMPUTER.

Please send me ______ copies of *LIST* at $12.50 a copy plus $2.00 each for postage and handling. (Tax will be added where applicable.)

☐ VISA ☐ MasterCard (Interbank No. ______________)



Card No. ______________________ Exp. Date ______

Signature ______________________________

Print Name ______________________________

Address ______________________________

City ______________ State ______ Zip ______

Send to *LIST,* Redgate Publishing Co., 3407 Ocean Drive, Vero Beach, FL 32960.

Or phone, toll-free: **1 800 821-7700 Ext. 1110**

LIST™

The Software Resource Book For Personal Computer Users

Announcing:
The Intelligent Keyboard . . .

SMARTKEY II™

What SMARTKEY does is simple.

SMARTKEY enables you to reprogram and key on your keyboard to hold from 1 to 7,500 characters. That simple capability opens up a lot of possibilities.

Want programmable keys? How about 256 of them?

Are you addicted to a powerful word processor with multi-key commands, like Wordstar™? SMARTKEY enables you to execute a complex sequence of control codes with one fingerstroke.

Are you a chronic programmer? SMARTKEY enables you to load a favorite subroutine or macro into a single key and unload it as often as you want.

You can re-alphabetize the alphabet, renumber your numbers, redefine control codes, switch to the Dvorak keyboard, and make those dusty, unused function keys useful.

For just $60, SMARTKEY increases the IQ of every key on your keyboard.

SMARTKEY, for CP/M®-80 and IBM-® PC-compatible systems.

Lifeboat Associates
1651 Third Avenue,
NY, NY 10028 (212) 860-0300
TWX: 710-581-2524 (LBSOFT NYK)
Telex: 640693 (LBSOFT NYK)

Prices and specifications subject to change without notice. Prices F.O.B. New York. Shipping, handling, C.O.D. charges extra. Wordstar, TM Micropro. CP/M, reg. TM Digital Research. IBM, reg. TM International Business Machines. Copyright © 1983 Lifeboat Associates. Smartkey, TM FBN Software.

Supercharge your REAL-NUMBER-CRUNCHING with the 8087 math chip and Float87™

Float87 adds real speed and super accuracy to your floating-point real-number calculations. Float87 offers software support for the Intel-8087 Numeric Data Processor, which supplements the capabilities of the 8088 and 8086 processor families, providing fast and accurate real-number functions.

How fast is fast?

A benchmark program written in PL/I ran over 45 times faster with Float87, and accuracy increased by a factor of 300.

Why waste time?

If your interest is in engineering or scientific data processing, simulation or graphics programs, real-time applications, or simply fast and accurate real-number-crunching, there's no time to waste. Interfaces are currently available for Lattice™ C and PL/I-86, with more languages to come. A unique "patch" program is available which automatically modifies Microsoft Basic-86 to take advantage of the 8087 chip.

Float87, for MS™-DOS, PC-DOS, and CP/M-86™ systems.

8087 math chip available.

Lifeboat Associates
1651 Third Avenue,
NY, NY 10028 (212) 860-0300
TWX: 710-581-2524 (LBSOFT NYK)
Telex: 640693 (LBSOFT NYK)

Prices and specifications subject to change without notice. Prices F.O.B. New York. Shipping, handling, C.O.D. charges extra. Lattice, TM Lattice, Inc. IBM, reg. TM International Business Machines. MS, TM Microsoft. © 1983 Lifeboat Associates. CP/M-86,TM Digital Research. Float87, TM Microfloat.

PMATE™

The Premier Text Editor for Programmers and Advanced Writers of all Kinds.

PMATE is the next generation in sophisticated text processors, a customizable editor with a powerful macro language that enables you to create your own text-processing functions.

- full-screen editing
- single-key commands
- horizontal scrolling
- automatic disk buffering
- eleven text buffers
- "undo" facility
- dynamic on-screen formatting
- programmable keys

PMATE is the text editor to write text editors in.

The macro command facility is really a compact and powerful text processing language inside PMATE, complete with variables, numerical calculations, conditionals, loops, and comments. Macros can perform a wide range of tasks including on-screen arithmetic, alphabetizing lists, translating 8080 code, preparing personalized form letters . . .

You won't believe what PMATE can do.

Until you try it. So try it. PMATE users become PMATE enthusiasts, because PMATE is the text editor that grows with you.

PMATE
for the IBM® PC, other MS™-DOS systems, and all CP/M-80 systems.

Lifeboat Associates
1651 Third Avenue,
NY, NY 10028 (212) 860-0300
TWX: 710-581-2524 (LBSOFT NYK)
Telex: 640693 (LBSOFT NYK)

Prices and specifications subject to change without notice. Prices F.O.B. New York. Shipping, handling, C.O.D. charges extra. IBM, reg. TM International Business Machines. MS, TM Microsoft. PMATE, TM Phoenix. © 1983 Lifeboat Associates.

Feature

The Bottom Line Strategist Reviewed

by Joseph B. Rothstein

The Bottom Line Strategist from Ashton-Tate, best known as the publishers of the popular dBASE II, represents the third generation of spreadsheet software. The first generation consisted of products like VisiCalc and its clones — computerized versions of the paper-and-pencil scratchpad, complete with its inflexible layout, fixed cell size, and other deficiencies. The second generation, exemplified by such products as Microsoft's Multiplan, followed the same blank spreadsheet concept, while easing some of the restrictions on cell size, layout, and formatting.

But with the third generation spreadsheet, the software not only includes provisions for the entry and manipulation of data, but also the rules, equations, and models necessary to adapt the blank spreadsheet concept to a particular real-life situation — in this case, developing financial or marketing strategies for products or services.

The user of the Bottom Line Strategist (BLS) must simply enter a series of figures corresponding to key business assumptions, and BLS then presents a number of different views of the growth curve, breakeven analysis, potential for profit or loss, and other projections in tabular or graphic format, rather than just the numerical results associated with traditional spreadsheets.

For this evaluation I used a Z-80-based S-100 system running CP/M, two double-density 8″ disk drives, and printer. I used BLS to evaluate strategy for an actual small business, and also for a number of hypothetical business ventures and products.

Installation

The hardware required to run the Bottom Line Strategist is more than many individuals, and even some businesses, may have on hand: an 8080- or 8086-family CPU running CP/M, MS-DOS or CPM/86, 64K RAM for the 8-bit systems (128K for most 16-bit systems), dual floppy disk drives, 80-column terminal with cursor addressing capability, and 120-column printer with tractor feed.

While this program is not aimed at hackers on a shoestring hardware budget, a more economical approach to its implementation might have made a difference. In fact, I was able to run BLS on a 60K CP/M system, but with less RAM than that the program simply bombed without so much as an error message.

Given these rather stringent hardware requirements, the actual program installation is quite easy. After making a backup copy to use as the working disk, the user simply runs an install program, BLSINSTL. It presents a menu which includes a wide range of terminals, one of which was the Televideo terminal I use. Once the terminal choice is entered, the program configures itself, and *voilá*, that's it.

Documentation

The printed documentation is bound in a three-ring notebook divided into nine sections. The first five sections, separated by grey index tab pages, include the introduction, startup, tutorial, and application development materials. The remaining sections, separated by red index tab pages, offer background materials — glossary, description of the financial models, references, and an index. On-screen HELP facilities (covered later in this section) are keyed to the printed documentation, and are mutually reinforcing.

Ashton-Tate deserves generally high marks for the quality and classy presentation of its documentation. It is clear, concise and to the point, and presented in a straightforward fashion without unnecessary clutter. Many software publishers seem to be responding to widespread criticism of the quality of their documentation by offering more pages, rather than better writing and organization. This rap was often leveled at dBASE II, which had a huge, sprawling manual, but it is not the case with BLS. In slightly more than 100 pages, I found all that I needed to know about the product, its background, and its use. The clarity of its organization and its complete glossary made it a breeze to use, both during the learning period and as a reference.

The "Introduction" section contains a Table of Contents, lists of figures and tables, an overview of the system including hardware and software requirements, tips on using diskettes, and installation procedures. Each of these is only a few paragraphs long, yet each is complete, accurate, and easily understood.

A section called "Getting Started" describes, in four succinct pages, how to initiate the program, create data files, choose an action from the Master Menu, print out results, and correct data entry errors. With only a few exceptions which will be noted subsequently, the commands and syntax are so straightforward that the rest of the manual is practically superfluous.

The "Lessons" section offers considerable detail, covering more than 30 pages. I looked at this section only as an afterthought, for the briefer discussions found elsewhere seemed just as effective a way to learn the system as a step-by-step tutorial. While my understanding of business principles is a far cry from an MBA's, I found that I didn't really need to understand the whys and wherefores in order to use the program. What BLS does, it does in a straightforward manner, and its weaknesses and limitations don't lie in the quality of its presentation. But for those who might want it, each of the program options is covered, and its effects described.

I found the section called "Modeling Your Business" much more valuable than the lessons, since it covered the processes involved in using BLS in a more generalized fashion and let me dive directly into real applications, rather than wading through pat and sterile examples.

Highlights of this section include thorough discussions of business growth curve projections, models of revenues and outlays, marketing costs, and cash-flow, net present value and break-even analysis. Each of these critical and highly technical subjects is carefully and lucidly explained, and reading them made me wonder why my previous experience with economics and accounting has always been so complicated and uninteresting. For those who really want to understand the guts of BLS, other reference sections include a complete mathematical exposition of the program's financial framework and lists of source material which should be available in a well-stocked public library.

The careful approach to documentation evident throughout the manual extends to on-line help messages as well. Such help is available for each of the 31 required user entries, and generally offers enough information to get the job done, without filling the CRT screen with excess verbiage. In addition, each help message cites the page in the printed documentation on which additional information may be found. This is a refreshing innovation — it seems such a natural that I'm surprised I've never before encountered the technique in other software.

Using the program

Once configured for a particular system, calling up and using BLS is a straightforward process. Invoking the program without specifying a data file enables the user to study the pre-programmed lessons, while specifying a file at invocation allows the user to work with prior models or save data entered for subsequent study, modification, and reuse.

The Master Menu includes one data entry option for the key business assumptions, four options for display of entered data, reports, and program output, and three choices which only serve to take up memory and screen space. These are the Executive Overview — an on-line sales pitch which seems altogether unnecessary — a hierarchical chart showing the Business Model, which properly belongs in the printed documentation rather than taking up valuable RAM, and an on-line copy of the Non-disclosure Agreement. This stuff must be loaded with the program every time it's run, adding to an already annoying wait and occupying memory space that might be used for other, more worthwhile purposes. This complaint would not be worth such a lengthy note, except for my fear that this unwholesome sort of practice might become widespread throughout the software industry.

The first option listed in the Master Menu allows the user to enter key business assumptions, or change those previously entered. These fall into eight specific categories: growth, marketing and advertising, costs, productivity, inflation accounting, cost-of-capital, and pricing policy. All dollar amounts must be entered in terms of thousands of dollars, though this may not be the most appropriate unit for all businesses. I would have preferred to be able to choose the default or, alternatively, to specify the units as part of each data entry item. In this case, consistency of data entry is gained at the expense of flexibility.

BLS permits users who are unsure of what value might be appropriate to use a system default, a value which is likely to apply to a wide variety of business situations. Certain values must be entered by the user, however, and BLS will prompt the user for them rather than permitting a default. The program will also trap input which is irrational or contradicts previous entries.

During the input phase, one-letter commands allow the user to move the cursor to different fields or data-entry screens, reset parameters, or get on-line help text. While these are logically consistent (U = cursor up, F = former screen), I would just as soon use the cursor control codes used by WordStar and many other programs. Even dBASE II uses many of these quasi-standard codes, and it's a shame to have to learn another set just for this one program.

Once the assumptions have been entered, they can be reviewed and grouped into the same eight categories by selecting Master Menu option 2. These can alternatively be routed to the printer and saved in hard-copy form.

Master Menu option 3 generates a forecast and analysis of profitability and tax shelter for the proposed product or venture, based on the assumptions entered. This is presented by month in a tabular format, in "windows" of 12 months each. For projections which run more than a year into the future, successive data lines overwrite those currently displayed.

The user controls the window by pressing control-S to pause in the overwriting and control-Q to resume. For those with less than nimble fingers, this is an imprecise and wholly unsatisfactory approach, for it requires a keen sense of timing in order to freeze the window in a particular position. It would have been far better, I think, to use the "page-pause" approach, stopping automatically at 6- or 12-month intervals until the user presses a continuation key. But like many of my criticisms of BLS's operation, this is only a minor shortcoming.

As with other display options, these forecast and analysis tables can be routed to the printer in a straightforward fashion.

Master Menu option 4 graphically displays the business forecasts. This is one of BLS's strongest points, for it provides a crisp, compact overview of what results from all this number-crunching. A variety of bar charts allows the user to quickly get a feel for growth trends, cash flow and other business factors in a way that no table of numbers could.

Notable in this regard is a "zoom" facility, which allows the user to concentrate on a portion of the graphic display, adjusting the display units to focus on a portion of the overall picture and display it in greater detail. Any of these graphics may be printed out.

The quality of these terminal and printer graphics depends on the type of terminal and printer used. But even with a system which does not support graphics characters, the images BLS produced using ASCII characters was sufficient to convey the ideas involved. Similar displays using a graphics terminal or printer, while not significantly different, would probably be more eye-catching and impressive to look at.

►

Finally, Master Menu option 5 allows the user to generate a series of tabular reports covering key business forecasts and profitability analysis.

Conclusions

The Bottom Line Strategist breaks new ground in spreadsheet software, offering as it does a preprogrammed model optimized for a particular set of circumstances.

That is both its strength and its limitation. Despite the advertising hype, which might lead one to believe that BLS is appropriate for any business forecasting situation, the range of situations for which BLS is an appropriate tool is actually much narrower. It is oriented toward a corporate entity which is considering the feasibility of a single new product, service, or marketing strategy. To the degree that a user's situation differs from that standard, BLS is less than ideal.

For example, if the business is a sole proprietorship or a partnership, BLS is limited in its ability to deal with such a venture's special needs and peculiarities. And since BLS is oriented toward the analysis of only a single new product or marketing strategy, it cannot handle multiple products which share development and marketing costs, and whose interaction influences the overall profitability of the larger venture. Furthermore, the recently enacted depreciation schedule known as Accelerated Cost Recovery System (ACRS) is not implemented; as such periodic changes to the tax laws are inevitably made, BLS will need to issue updates or risk becoming outdated.

Despite these limitations, BLS would probably be a valuable tool for most start-up companies or typical, small businesses. Although it might not serve the business's needs precisely, it forces the business person to think carefully about the assumptions behind the business. Most small businesses neglect this critical aspect of planning, if in fact they plan at all. No established corporation would dream of embarking on a new product introduction, service, or marketing strategy without the kind of market research and consideration of strategy that BLS encourages. Simply by using BLS, the business person must think in these terms.

Though the Mom & Pop store may not have at its disposal the research facilities of a Ma Bell or General Motors, using BLS is like peeking into one of these giants' business manuals. Approaching the business in an organized, forward-looking fashion — rather than making haphazard decisions or responding to each crisis as it arises — can only benefit the bottom line, and in business, that's all that matters. ■

TABLE 1
Facts & Figures

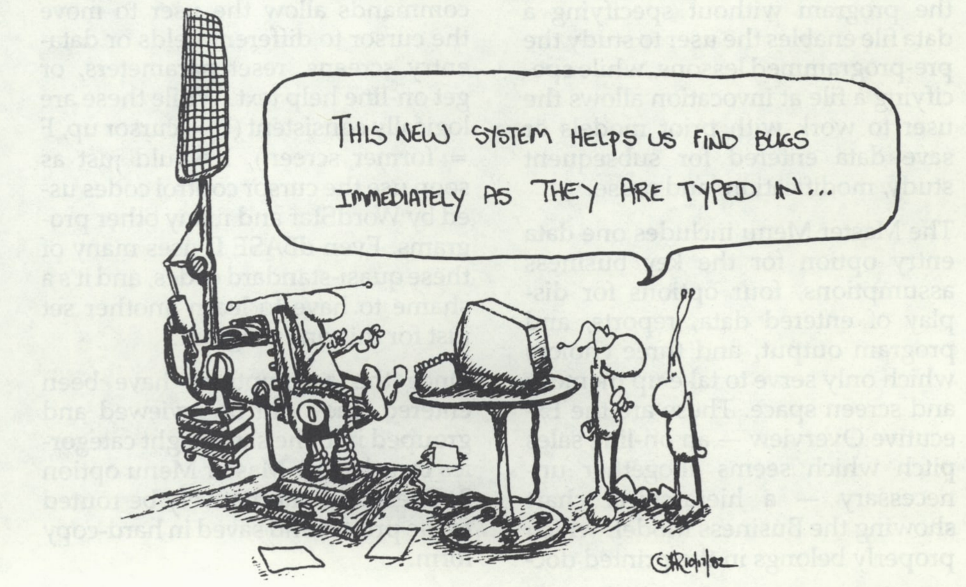
Package and Version Name:	Bottom Line Strategist, Version 1.0
Supplier:	Ashton-Tate 10150 West Jefferson Boulevard Culver City, CA 90230 (213) 204-5570
Suggested Retail Price:	$400
Operating Systems:	CP/M version 2.2 MS-DOS version 1.1 CP/M-86 version 1.0
Memory Requirements:	64K RAM specified (8-bit systems) 56K Ram (Apple II CP/M) 128K RAM (MS-DOS and CP/M-86) 256K RAM (Victor 9000 version)
Disk Drive Requirements:	Dual floppy disk drives — 160K each Hard disk drive optional (MS-DOS version)
Disk Formats:	8" standard, plus various 5¼"
Documentation:	Printed manual in 3-ring binder, plus on-line help messages
User Skilled Required:	Novice computer user; minimum understanding of essential business practices
Update Policy:	Guarantee of current version at time of purchase. Registered users will be notified of subsequent updates and purchase price

TABLE 2
Qualitative Factors

	Rating*
Documentation	
organization for learning	6
organization for reference	6
readability	7
includes all needed information	6
Ease of Use	
initial start up	6
operator use	5
adaptability	4
Error Recovery	
inadequate system memory	3
from input error	5
restart from interruption	6
Support	
for initial startup	5
for system improvement	5

*Ratings in this table will be in a 1-7 scale where:

1 = clearly acceptable for normal use
4 = good enough to serve for most situations
7 = excellent, powerful, or very easy depending on the category


THIS NEW SYSTEM HELPS US FIND BUGS IMMEDIATELY AS THEY ARE TYPED IN...

Feature

Computers And Organizations

by Jon Wettingfeld

Part 2: Automating The Office

In our last article we tried to pinpoint general areas in which the "new technology" would have greatest impact in modern management groups. The general conclusion, drawn from work by people both inside and outside the information sciences field, was that technology, from mainframes to micros, could have a significant impact on productivity if it improved the flow of information on which decisions are based. In addition, because of savings connected with automating the office, and the potential for organizational growth connected with better communications systems, the "forecast" for increased use of automation as a productivity tool on many levels, managerial, executive and even secretarial and small business, seemed very bright indeed.

Thus it isn't suprising to see projections for the business generated just by micro computers to be estimated as high as 11 billion dollars by 1995 (in the U.S. alone). On the other hand, world market potential in micro sales, according to one source, may be reaching somewhere around 67 billion dollars by 1986. There can be little doubt that the benefits of office automation are being recognized — people want the productivity enhancements that automating will bring. In fact, some say that the ONLY way for a business to keep ahead of the competition is to incorporate these systems, and that those that don't will go the way of the vacuum tube!

This month we will look at some of the specific trends in this very dynamic area of office automation. The focus will be on the micro computer. Again, we'll try to spot the possible long-term outcome of the automation "revolution" in the office. But this time we will be looking at specific changes and market trends. These will be in the areas of word processing, applications packages in general, database management in micros, and lastly the impact of automation on the workplace as a whole.

What's In A Word? (Processor)

Word processing, along with the mainframe, was one of the first harbingers of the eletronic revolution in the business world. During the 1960s, large businesses began to use computers for functions such as inventory, payroll, and accounts payable. Soon after such electronic equipment became cost effective, the word processor was at the forefront of the campaign to automate. Today, change in this pioneering technology may again augur future trends in the office. A look at the word processing market suggests that it may indeed be a microcosm, representing the kind of transition occurring in much of the data processing world. For, as costs are cut and people expect more from systems, there will be long-range effects on potential capabilities of software and hardware. And word processors have been termed one of the significant segments of the low end/high tech hardware and software market. As such they are likely to be affected significantly by market forces, along with other important productivity tools such as spread sheets, graphics packages, and data base systems for micros (more on those later).

As cost cuts in software and hardware bring micros within reach of even the smallest businesses, and as people continue to expect more powerful and more user-friendly software packages, some see a corresponding shift in the word processing field. Some have suggested that this trend may lead to the eroding of the distinction between word processing in micros and word processing in a dedicated environment.

The original distinction between dedicated and non-dedicated markets may well have been the result of the hardware capabilities of the early micro computers, in contrast to the dedicated systems. Micros initially had slower processors, smaller memories, and lacked the 80-column screens necessary for word processor use. Micro computer software, on the other hand, lacked important capabilities, notably the ability to use the "soft function" keys. "Generic software" also lacked the simple, powerful commands of the dedicated software. But a trend towards improvement of such software has led to a more powerful product. While some say that micros with both data and word processing abilities can never truly usurp the use of dedicated machines in offices where MOST of the automated work is word processing (e.g., a law office), others feel there is reason to believe that the new software capabilities may lead businesses with "casual" word processing needs to select general purpose micros in the low end of the market. This would be in lieu of going with a dedicated machine, and would divert purchasers of low end WPs to the micro market. Thus, more powerful software may be pushing the hardware market towards an integrated approach to automating office work.

Conversely, some cases exist where hardware itself is being designed with a dual function in mind. Sterns Computers recently introduced what it called a "true dual function machine." It is supposed to achieve equal capability in both the WP and DP areas via careful design. Another response to the drawing away of the "casual user WP market" (defined by one writer as people who use a machine for WP less than 30% of the time) from dedicated machines has been to enhance DP and other capabilities of the dedicated machine itself. Thus, tying like systems together, mass storage capability, data base management, data processing, calendaring and electronic mail are features being engineered into some otherwise dedicated systems. In short, in both the hardware and software of the word processing field, we ►

see attempts to merge the capabilities of two formerly isolated markets. However, consolidation seems not to be limited exclusively to the WP/DP rubicon. Perhaps one of the most exciting changes in low end automation has been the increased availability of economical data base management programs for micros in lieu of discrete software packages.

Databases: All Things To Most Applications

An important form of segmentation in the micro computer world has been "segregation" of software applications packages. For example, Malcolm L. Stielfel, in a recent issue of *Business Computing*, points out that various employee packages, payroll, tax withholding,etc., may have to be updated repeatedly, because they contain separate, often redundant, lists of employee names, addresses, telephone numbers, etc. An additional drawback is that reports in these packages are usually generated in pre-determined formats, allowing little flexibility. As a solution to these problems, Stielfel cites many instances in which microcomputer users have "thrown out" standard business applications in favor of a different approach.

Instead of distinct categories of software such as payroll, receivables, order entry, etc., an increasing use is being made of databases to handle all these functions. Interestingly, it may be a measure of the phenomenal rate of change in the microcomputer market that only three years ago, C. Roger Smolin of San Diego Business Systems could write, "I do not know of any database management systems currently available for small business systems...up to now they have been associated mostly with large scale systems...and mini computers." Contrast this with Stielfel's recent listing of more than seventeen different database packages currently available for micro environments!

Of course, there are still significant limitations to how powerful micro database systems are, as well as in how they are actually organized. For the most part, the powerful hierarchical and network type of database systems are still relegated to mini and mainframe use because of their requirements for memory and processing power. Instead, the most commonly used system on micros is the relational system. These have limitations, but also powerful "electronic cut and paste" capabilities. The user can easily cull information from files, creating completely new ones, or joining separate pre-existing ones. Mini/micro differences aside, the significance of database systems is in the unified approach to information management. The inherent advantage of the database system lies in its flexibility. Instead of separate applications and separate types of files for each type of application, a DBMS relies on one large file or database. Data items or data sets may be linked or re-linked to form different views of the information stored. Instead of worrying about the logistics of linking unrelated files, or having a programmer re-write applications to accommodate the addition of new types of information, one can simply add new data to the single existing data base when needed. This is readily done through the use of what is usually a very English-like applications language. This language allows ready deletion, insertion, and modification of data items. It also allows reports, screen retrievals of information, etc., on either a flexible case by case basis, or in the standard forms needed in regular business applications packages. It is in this sense that the database approach has been called "the ultimate programming tool." It does away with the need for complex programming and at the same time increases the user friendliness of the machine — the software takes on more functions. This is true both because it can perform in place of many applications, and because it takes care of more of the programming chores for the user. Interestingly, in databases as in word processing, we see again an increase in the power of automated software. With the move of the DBMS into the micro environment, there is, at the same time, a move towards a more versatile, "integrated" micro system. David R. Roman, writing for *Computer Decisions*, reports a similar move towards data software versatility. This involves the interchangeable use of micro and mainframe data with a micro spreadsheet package. Other writers have pointed out that manufacturers of applications are increasing compatibility between otherwise separate packages.

Integration: From Word Processer To Work Station

Another extremely important "integrated" approach to automation may be the connection of various micros, smart and dumb terminals, all connected to outside mainframes as well as to each other. Limited Area Network Systems (LANS) and other communication systems will have a profound impact on office automation. They will be the subject of a future article. (See also September's *Lifelines'* editorial.) But looking generally at the subject of increased automated flexibility, it is possible to see parallels between the hardware developments and the software areas already mentioned. For example, according to Sandra Majorowicz of *Office Products Dealer* magazine, Wang Laboratories director of office systems John Thibault has projected the following scenario. He sees a merger taking place which is similar to the limited one we saw between WP functions and general use micros, but on the broader level of combining different machines. This will involve the "blend of high and low end 'high' technology to achieve an integrated system to meet business needs." In his view, most office functions, calendaring, scheduling, electronic mail, messages, database management, document traffic and word processing, will all be automated (though he foresees specialization in the individual work stations based on job category). To accomplish this he foresees an "economical" blend of mini, PCs, and mainframes combined in a LANS. John O'Keefe, a product group manager at Digital Equipment Corp., has taken up the torch for system integration. He suggests that eventually word processing and data processing will merge in the office as one overall system. This echoes the philosophy voiced by companies like Sterns, cited earlier. Reinforcing this view, and espousing a truly futuristic vision of the outcome of the high technology revolution in the workplace, Vincent Giuliano has proposed the "work station" as the ultimate denouement of the automating process.

Writing in the September 1982 issue of *Scientific American*, Giuliano suggests several overall trends. One is the shift from paper-dependent offices to organizations utilizing terminals with access to mainframe databases, communications networks, etc. He feels that by 1990, 40% to 50% of American workers will be using terminal equipment, with about 38 million terminal-based work stations in offices, factories, and schools. (This does not include home terminals.)

In addition to increases in number, an increase in sophistication of these personal work stations is expected to occur. Unlike many of the present automated stations, the new ones will be MULTI-PURPOSE in nature. A current example cited of this is the Xerox 8010 star work station, having standalone capabilities as well as being usable in a LANS. He further suggests that this technology will lead work organization away from the "industrial office" layout, in which tasks are partitioned or segmented in an assembly-line fashion (and in which errors may be accumulated with almost as much efficiency as the work that is being completed is accomplished) to the "information age office." This new "evolutionary stage" allows the "integration" of many service tasks (and presumably the systems by which they are produced). This is accomplished at terminal work stations which are combined with effective communications and a continuously updated database. The arrangement produces "people centered" work, which, instead of small, incremented, repetitious tasks divided among departments, utilizes a customer service representative. This person handles all customer-related activities for a few customers. The individual worker, in effect, becomes the account manager. While not all feel that such integration via one system is feasible (one writer has pointed out that businesses, when automating, use devices by many different manufacturers — that it is not practical for a single company's technology to supply all of a business's needs), the prevailing view seems to be that integration of functions via automation is possible and desirable. For the individual micro, the move towards compatability between micro computer applications, whether by databases or merging of other software functions, seems to reflect this at the "low end" of high technology. And this is supported by Giuliano's projections for both low and high end automation. In the long run, he says, the economics of fast, cheap electronic communications, based on lower and lower costs, is a factor in determining these trends. As prices of processors and memory continue to fall, such equipment will be available "as needed." This will have several major effects. As suggested earlier, it will reduce the costs of record duplication, as well as a significant business expense, managerial time. As pointed out in last month's article, a substantial improvement in productivity is also involved in reducing intra-organizational information distortion and loss. Since the office is the locus of decision making in modern administrative organizations, these improvements can only result in more cost-effective decisions.

Lastly, and most startling, Giuliano suggests that integrated and WIDELY AVAILABLE information access may change the organization of the workplace itself. If such access approaches the universality of the "telephone's availability" work may no longer be bound to a single location. This means that individuals may be able to organize their own time, working where and when they want to. Giualiano calls this the advent of the "virtual office," and certainly anyone who has ever "brought work home" via a portable terminal/modem or PC can accept the possibility of its eventual existence.

To sum up, underlying the explosive office automation market may be several factors. People may be recognizing that better information equals better decision making, and that electronic communications cut costs both in paperwork and management time. Other factors in the market expansion may be the decline in the price of data processing equipment. Whatever the reasons for the burgeoning market, the industry itself seems to be moving towards producing more powerful software and more versatile hardware — an integration of many functions into some of the smallest equipment ever seen. (Giuliano points out that the memory and computing power found in a 1982 16-bit portable is equal to that of a mainframe in 1952!) This integration seems to be taking place in a number of areas. Ultimately, increases in technological capabilities in information processing may lead to a dramatic reorganization of the work place. ■

Change of Address

Please notify us immediately if you move. Use the form below. In the section marked "Old Address," affix your *Lifelines* mailing label—or write out your old address exactly as it appears on the label. This will help the *Lifelines* Circulation Department to expedite your request.

New Address:

NAME

COMPANY

STREET ADDRESS

CITY　　STATE　　ZIP CODE

Old Address:

NAME

COMPANY

STREET ADDRESS

CITY　　STATE　　ZIP CODE

Moonlighting With A Computer: Where's The Right Opportunity?

by J. Norman Goode

Does the prospect of using a computer as the center of your own business venture scare or excite you? How about both? Well, if you have been exposed to television, newspapers, and magazines for the last year or so you know that the personal computer has become the new toy on the block. The media may even have you convinced that you are depriving yourself and your children if you don't have the latest in personal computers in your home. Maybe so — maybe not. Before we decide that question and look at the prospects of using a computer in a home business, let's look at some of the feelings you, and many others, may encounter when confronted with a computer.

For the last 20 years or so, the computer has acquired a mystique of its own. You're probably much like me. We grew up in times of concern about The Bomb, satellites, the space race, and the threat of "what has science wrought." I think that I have seen virtually every Grade B science fiction film of the era. I remember the concept of the computer as a great intelligent machine with blinking lights, whirring tapes, and a metallic voice. The Computer was always super smart, a dictator, and threatening. I think that, largely because of these films, people have held onto the wrong concept of the computer. I have encountered many folks who are afraid that they will cause the world to destruct if they press the wrong key on the keyboard!

The reality of the computer, though, is that it is nothing but a TOOL, pure and simple. If you can imagine the value of the invention of the wheel to all of the modern conveniences we now know, then you can appreciate my view that the personal computer is the modern version of the wheel. It wasn't too many years ago that a computer required a room of its own, complete with air conditioning, a special crew to run it, and a price tag of hundreds of thousands to millions of dollars. Today, for an investment of a few hundred to a few thousand dollars, anyone can have the same computing power sitting on his/her own desk.

So then — your evaluation of whether you should use a computer for your own particular business pursuit should depend more on the use of that marvelous tool to increase your own productivity and potential for profit-making than as a gee-whiz gadget that you must have because everyone else (or so the advertisers would like to think) has one. With this view, the computer becomes a super bookkeeper; a fantastic calculator; a grand way to write correspondence; and your "right arm" as an employee who doesn't require pension plans, medical insurance, withholding taxes, or Social Security. It doesn't take breaks, vacations, or call in sick, and is a slave to your every command, if you so desire.

So much for the computer as an ordinary tool to help keep books, write letters, etc. You could accomplish the same tasks with a pocket calculator and a typewriter. These tasks might take you a little longer if done manually, but would not require the investment needed for a personal computer. The real beauty and potential of using a personal computer to generate income is that this same tool can be used as an EXTENSION OF YOUR MIND! This means that the limit of what you can do to utilize this wonderful tool is truly infinte. In the three years of publishing *Micro Moonlighter Newsletter,* I never cease to be amazed by the creative genius shown by entrepreneurs who have taken, literally, the challenge to use the computer as an appendage to their minds in the pursuit of their own business goals. An associate of mine once remarked that the vast majority of business' ideas for use of the personal computer has yet to be discovered. We will surely see thousands of new approaches to accumulating wealth in the next few years as people, like you, create new services and products based upon the use of their personal computers.

The opportunities: Today and tomorrow

The most often-asked question is usually "What can I do to earn money with my computer?" When I give the standard answer of "anything you want to do," I usually get a glazed look and silence in return. Sometimes it takes concrete examples of what others are doing to start one thinking of how to approach the new role of micro-entrepreneur (my term for the business adventurer using a computer). The following represents what the "hot" areas are today for micro-entrepreneurs:

The area with the most potential for earning truly mind-staggering sums of money is software development. Software is the term for the programs which turn the computer from a dumb machine to a useful tool. Predictions are that the need for home computer software will grow by 70% through 1987. Software for business applications is expected to become a $7 billion market in that time period alone. Learning to program a computer isn't as difficult as you might think. Again, we have that concept that programmers are whizzes at math and physics. Nothing could be further from the truth. It is not something that you can do overnight, but with the help of the many manuals available at popular book stores today, there's no reason why the average intelligent person should not be able to read and practice the art of programming and become proficient enough to write salable software. Royalty rates given by major software publishers vary and are usually from 15% to 20% for each copy sold. This doesn't sound like much, but when we talk about thousands of copies of a product being sold, the income potential mounts.

Another area of current interest is word processing and all of its variations. On a very simple level, word processing is simply a typing business using a computer instead of a typewriter. The income potential comes from using the expanded capability of the computer to offer services that a typing service cannot. A typical service is the production of "boilerplate" letters. A business may want to make a mailing to several hundred of its customers. The business may want to present the same sales message to every one of them, but it also wants a "personal touch." You, with your word processing software, can first enter the business' customer list into the computer. Then, because of the capability of the computer, you only type the basic letter one time. When you call upon your electronic employee to do so, it will then type an individual letter for each customer, personally addressed to each!

The income potential here is also very good. In a survey I conducted recently, I found that per page rates for word processing service businesses ran from 75 cents to $7.50 per double-spaced page. The higher rate was found in college communities where there is much call for the typing of dissertations and theses. There are many manuals available which give good advice for starting your own word processing service. Should you not be able to find them locally, drop me a note at *Micro Moonlighter* and I'll be glad to furnish some sources.

The third area for income potential these days is consulting. A consultant is an expert in his/her particular field of work. Many people would be surprised to learn that they, by virtue of doing a particular job for many years, have the potential for becoming an independent consultant in that particular field. Schools and colleges are doing a poor job of training these days and businesses are always in need of someone who has the experience to bail them out of a bad situation. Enter the consultant to save the day! Armed with a computer the consultant is able to analyze the situation and offer a solution to the problem.

Micro consultants are the ones who are making it big these days. These are people who have accumulated good knowledge in their field and have then become well acquainted with their computer. They have selected the best software available to help them in their endeavor and have presented themselves to potential clients well. Here, again, the income potential is open ended. I have a friend who planned his jump into consulting for about a year. When he made the plunge, he had all of his tools in order. Since beginning his practice, he bills out $600 a day for his advice!

Then there's the broad area called "The Information Services Industry." Yes, publishing a newsletter as I do is one portion of this industry. And, yes, it is profitable. Whenever a new technology or area of opportunity arises, there also exists the opportunity to create a business of informing others of the new industry or technique or area of opportunity. The field is now expanding at a tremendous rate. There are new information databases available every month. These large computer data banks are available to the personal computer owner via a hook-up through ordinary telephone lines.

With these information resources available to the cottage worker, the area of specialized research for profit is expanding. Companies are willing to pay handsomely for specific information which can be garnered from access to these databases. All that is required to operate this business is your computer, a good degree of organization, and the ability to string some words together in an orderly manner (popularly known as writing). Entrepreneurs are charging anywhere from $200 to several thousands of dollars for each report generated, depending on the amount of research required and the amount of detail in the report.

The above items represent only the current "hot" areas where people are earning a good income. Other areas represent such opportunities as computer camps (for adults as well as children), schools to introduce the computer to others, bookkeeping services, call-back services for doctors and dentists, stock market analysis for investors, and a world of others. One particularly creative subscriber programmed his computer to generate word-find puzzles while he slept. He then only had to tear the puzzles from the printer each day and submit them to publishers of puzzle books for purchase. He called it his "money machine" because it literally made money for him while he slept!

The future? Well, the future is upon us at every turn. I foresee the information industry as expanding. The new "toy" of robotics should prove to be interesting. The design and use of "personal robots" is at the same stage that personal computers were 10 years ago. Most of those involved are hobbyists and tinkerers. Look at where that got us in personal computers!

Again, let me emphasize that the future of using a computer for your own moonlighting or full-time venture is in YOUR OWN MIND! Remember that the computer is a tool composed of a blank sheet of paper. It is up to you to fill in that blank sheet and use the tool for whatever job or opportunity you have in mind.

What does it take?

The skills for using your computer for your own business venture are the same skills you would need for any business startup. I would add the following to those skills and needs, especially for a computer-based business:

First, whether you own a computer or not, don't buy a lot of computer hardware and/or software and then decide what you want to do with it. Before doing anything you should read, read, and then read some more. Subscribe to computer publications — buy books about computers and their use. Find out what is currently being done in the field! The books, newsletters, and magazines are tax deductible and can save you much grief and money.

Second, plan to throw away your clock in the first year. This rule applies to any business startup, computer or not, and the reason should be obvious. You are only going to get as much return from your business as you put into it. It will take a lot of your time to promote and nurture your undertaking. My guideline has always been that no one is going to knock on my door asking to subscribe to my newsletter. I must DO SOMETHING to make my business a living thing. If I take care of it, it will take care of me.

(continued on page 23)

Good news, Your ship

Ashton-Tate proudly announces an unforgettable three-day informational extravaganza aboard the historic Queen Mary in Long Beach, California: The first dBASE II RunTime™ Conference.

For dBASE II old salts. And landlubbers, too.

If you're an old hand at dBASE II program publishing, our on-board entrepreneurs and experts will show you how to expand your base of operations.

And even if you've just written your first dBASE II command, we can help you launch a whole new career with dBASE II and RunTime, our applications development module.

You'll get invaluable business and technical insights from people like Ashton-Tate president David Cole. Wayne Ratliff, the man who gave the world dBASE II. SoftSel president Robert Leff. Meir Geller of Fox & Geller. Representatives from Apple and Northstar. And many, many others.

So don't miss the boat.

Over 150,000 software developers and end-users have found dBASE II to be the solution to just about every conceivable information handling need, in business, scientific and educational applications.

Join us on the Queen Mary and let us share their experiences and ours with you. It'll put some real wind into your sails.

Ashton-Tate, 10150 West Jefferson Boulevard, Culver City, CA 90230. To register, just send the coupon or call (213) 204-5570.

GUEST SPEAKER
Professor E. A. Feigenbaum, Stanford University, co-author of The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World.

OVERVIEW
Getting the most out of vertical markets.

BUSINESS BASICS.
Surviving the start-up.
Legal issues.
Advertising and publicity.

TECHNICAL WORKSHOPS
Basic programming techniques.
Advanced methods and short-cuts.
Maximizing dBASE II power.
The internals of dBASE II.
New features for dBASE II.
Documenting your applications.

THE DISTRIBUTION PROCESS.
Assessing marketability.
Choosing and using distribution channels.
How to talk to dealers.
Pricing and promotion.

DEVELOPING AND MARKETING APPLICATIONS PROGRAMS.
Entrepreneurs tell all.
Selling to OEM's and corporations.

MARKETING SPECIALIZED APPLICATIONS.
The financial market.
Products for professionals.
Record and library management.
The non-profit sector.

dBASE II® fans: has come in.

I don't want to miss the boat, so here's my registration:	
Name ______	
Company ______	
Address ______	
City/State/ZIP ______	
I would like ____ registrations at $350 per person.	☐ Check
	☐ Money Order
Card # ______ Expires ______	☐ VISA
Signature ______	☐ MasterCharge

Send to: Ashton-Tate, 10150 W. Jefferson Blvd., Culver City, CA 90230.
Or call (213) 204-5570, extension 300

ASHTON·TATE

dBASE II RunTime Conference
November 10-12, 1983.

►

COBOL-80 Reviewed

by John Blanton

Introduction

COBOL is usually not the first programming language a home system owner adds to his disk files but it *is* one of the first computer languages ever, and its position in the business world is firmly seated. The target application of COBOL is strictly business; accounting, inventory, payroll and such, and more than any other popular programming language it attempts to offer the user a completely English environment, isolating the business programmer as much as possible from the intricacies of the computer. However, in doing this it has acquired such a wordiness and such an extensive set of key words that it has become somewhat unwieldy for the casual user.

All of this notwithstanding, COBOL is still the most popular language of the so-called "data processing" world, and predictions of its replacement by Pascal are very premature. COBOL has gained so much momentum throughout the twenty-five years of its use, there is so much existent software written in COBOL, and there are so many programmers already trained in its use, that there is little incentive for business data processing departments to go shopping for another language.

COBOL-80

Microsoft's 8-bit version of COBOL has been around for several years, and it is presently up to version 4.65, running under CP/M-80.

COBOL-80 is a fairly extensive implementation of the COBOL ANSI standard, containing nearly all of the Level II implementations (there are two levels of implementation under the ANSI standard, and II is the highest) for the Nucleus and Table Handling modules and for the three file-handling modules (Sequential I/O, Relative I/O and Indexed I/O). The Interprogram Communication and Library modules are implemented to Level I, and the ANSI Debug and Report-Writer modules are not implemented at all.

However, COBOL-80 does provide several extensions to ANSI standard COBOL. These include the IBM Debug facility, interactive screen control and the COMP-3 data format (for packed numeric data).

COBOL-80 has all of the standard COBOL mathematical features plus the table-handling features (including search operations) and the file-handling features found in other languages. Of great interest, however, are COBOL's indexed file handling facilities, and in this area COBOL-80 is well represented. Although COBOL-80 does not allow multiple keys in accessing indexed files (as required in a *full* ANSI COBOL implementation), it does provide all of the standard indexed file access features including: 1) sequential, random, and dynamic (intermixed sequential and random access operations); 2) read, write, rewrite (replace a record after reading), start (find a record by use of a key field) and delete; 3) conditional operations including INVALID KEY (what to do if keys do not match) and AT END (what to do when end of file is reached).

Extensions

In addition to the standard COBOL features, there are some extensions provided by COBOL-80 that will be appreciated by many users:

DEBUGGING. The interactive debug facility should be very useful in checking out new programs. It is invoked by linking the file DEBUG.REL with the program object modules, and it allows the user to access program code (by referencing program line numbers) and program data items (by name). To do this the debug facility makes use of a file called <name>.DBG, where <name> is the name supplied in the PROGRAM-ID paragraph of the COBOL program. The debug facility will be entered automatically when the linked program is executed.

With the debug facility the operator can perform the following:

Display a help list of debug commands.

Display the address of a data item referenced by name (subscripted data items and variables not formally defined in the DATA DIVISION cannot be accessed).

Display and change the value of data items. (An effective field-feature is incorporated for this purpose.)

Set up to eight simultaneous breakpoints, list breakpoints, change breakpoints, delete breakpoints.

Begin or resume execution at the current breakpoint or at any line number.

Execute the program one step at a time.

Terminate program execution.

Enable and disable a line number trace mode.

SCREEN CONTROL. Screen controls are used in interactive data entry mode during program execution and are invoked by first defining the screen format in the SCREEN SECTION within the Data Division of the COBOL program, then subsequently referencing the screen name within an ACCEPT statement. The extent to which a particular application can use the screen controls is determined by the attributes of the display used with the computer, but the controls allow for the following:

Clearing the entire screen

Erasing a line

Cursor positioning

Sounding the 'bell'

Highlighting or blinking certain fields

Left-justifying, right justifying data within screen fields

Automatic termination of field input when a field is complete

Securing a data field by not displaying its contents

By providing a means for efficient and compact definition of interactive screen format controls this feature facilitates the production of sophisticated data entry routines. It is important to note that simple ACCEPT and DISPLAY statements can be used for console communication without requiring a formal SCREEN SECTION definition.

COMP-3 DATA FORMAT. The COMP-3 data format allows the encoding of two decimal digits of data within a single byte and, thereby, provides for more efficient use of disk storage in data files.

Customizations

COBOL-80 makes provisions for certain user customizations, both to accommodate individual preference and to tailor systems to particular hardware characteristics.

CRT DRIVERS. To allow users to adapt systems to specific terminals, Microsoft supplies a variety of terminal driver modules both in assembly language source form and in already-assembled object modules. The user is expected to select the driver that matches the specific terminal in use, place the object file on the same disk with the other COBOL object libraries, and to rename it "CRTDRV.REL." Then, when an executable program file is generated using the Link-80 linker, the CRT driver will be automatically included. If none of the supplied terminal drivers exactly match a specific terminal's requirements, then the assembly language source file of one of the driver modules can be edited, and the resulting module can be assembled and used. Drivers for the following terminals are supplied:

ANSI standard	Lear-Siegler ADM3-A
Beehive 100, 150	Microbee 2
Cromemco 3101, 3102	SOROC IQ
Hazeltine 1500	Heath WH19
DEC VT52 (no cursor on/off or highlight/blink)	ADDS Regent Terminals (40, 60, 100, 200)
Perkin-Elmer	Zentec Zephr
Intertec Superbrain	IMSAI VIO

TAB STOPS. If tab stops are used within the COBOL source code the compiler supplies the spaces needed to reach the predefined tab positions, namely 8, 12, 20, 28, 36, 44, 52, 60 and 68. However, Microsoft supplies information for altering these defaults within the COBOL compiler (DDT is needed to perform these alterations).

COMPILER LISTING PAGE LENGTH. Once again information is supplied to allow the compiler to be patched for this change.

RUNTIME DAY, DATE, TIME, LINE NUMBER. The compiler supplies default runtime data (the default date is the compiler release date) and user terminal line number (assumed to be "00" for single user systems. To allow use of system utilities that can supply actual time, date and user number information, Microsoft supplies extensive information for building and installing the required interface modules. This customization requires writing the interface module in assembly language, assembling the module and adding the module to the runtime library using the LIB-80 utility.

Operational aspects

Microsoft COBOL-80 resides on disk, not as a single file but as a primary COM file plus four overlay files. The compiler is invoked by running the file COBOL.COM which, in turn, loads and runs the overlays as they are needed to process the successive stages of compilation. This method allows a single, relatively small segment of memory to serve for each of the various overlays, thereby making the most memory possible available for program file processing.

The COBOL command line is typical of Microsoft compilers and assemblers. It allows the source file and optional print and object files to be named in the single command line, which can include several compiler "switches." A typical command line might look like this:

```
A> COBOL
B:OBJECT,B:PRINT=B:SOURCE
```

In the above the B drive is specified for each of the object, print, and source files. If the drive specification is omitted for any of them, then the currently logged drive is assumed for that file. Note, also, that no file type has been specified here. Type specification is optional for each of the three files, and if it is omitted the defaults of REL for object files, PRN for compiler-listing files and COB for source files are assumed. Of the three files, only the source file need be specified. For example, the command line:

```
A>COBOL =D:SOURCE
```

will read the source file SOURCE.COB from drive D and will create the object file SOURCE.REL on drive D. No listing file will be produced. The line:

```
A>COBOL ,D:LIST=D:SOURCE
```

will produce the compiler-listing file LIST.PRN on drive D. No object file will be produced.

One or more compiler "switches" can be appended to the command line. These switches take the form of a "/" followed by the switch designation, and they are:

/R	Force generation of object file
/L	Force generation of list file
/P	Allocate additional stack area for the compiler
/D	Omit generation of debug information
/X	Generate object file linkable with the runtime library, RUNCOB.LIB
/Fn	Generate FIPS flagging messages in the list file where "n" controls the level of flagging features listed. (This allows the user to assess the compatibility of the COBOL source file with various ANSI levels of COBOL implementation.)

After one or more type REL relocatable object modules have been generated using the compiler, they may be linked with the runtime system library (if the /X switch was invoked) and with the CRT driver module (automatically if required by the program) and built into a type COM executable program file.

The "executable" program file produced is not machine code but is a pseudo-code that is interpreted by the runtime system. Depending on how the program is compiled, the runtime system can be either included in the COM file, or else the runtime module RUNCOB.COM will be invoked by the program file when it is loaded. The file, COBLOC, specifies for the program the drive on which RUNCOB.COM is located. Once executing, the COBOL program can invoke other COBOL programs as overlays from various sys- ►

tem disk drives, thereby allowing quite large COBOL systems to be run in CP/M's maximum 64K memory. RUNCOB.COM remains resident during chaining between COBOL programs.

In order to use the services of the interactive debug facility the file, DEBUG.REL, must be explicitly included during the linking process.

It is interesting to note the consequences of various program configuration options. In a test, one COBOL source program file was built into various optional configurations by successively compiling it using the different compiler switches and then linking the required modules to produce an operational program. The program is quite representative, since it uses screen input and output plus printer output and multiple disk file access.

In the first case the program was compiled with the /D switch to eliminate debugging aids within the object module. The program was not linked with COBLIB.REL, the relocatable version of the runtime module. In the second case the /D switch was not used (causing debug reference information to be included within the object module), but the program was not linked to DEBUG. REL (and was, again, not linked to COBLIB.REL). In the third case, the object module from case 2 was linked to DEBUG.REL (but still not linked to the runtime module). Cases 4 and 5 both involved linking the program object module to COBLIB.REL, but the object module in case 4 was built using the /D compiler switch. Finally, case 6 shows the result of building a program module using COBOL-80, version 4.01:

Case	Description	Program Disk Image	Size
1	/D switch, not linked to COBLIB.REL	12389	bytes
2	Not linked to COBLIB.REL, no /D	12725	bytes
3	Same as 2, linked to DEBUG.REL	21933	bytes
4	Linked to COBLIB.REL, /D switch	29365	bytes
5	Linked to COBLIB.REL, no /D switch	29701	bytes
6	COBOL-80, 4.01 (linked to COBLIB.REL)	26406	bytes

Note the difference between 1 and 2. Over 300 bytes are eliminated from the program disk image with the use of the /D compiler switch. Case 3 shows the addition of over 9000 bytes by linking to DEBUG.REL. Cases 4 and 5, compared to cases 1 and 2, respectively, show the increase of nearly 17,000 bytes in the disk image when the runtime module is linked. Finally, case 6 compared to case 4 shows that, curiously, the disk image was smaller under version 4.01, even though version 4.65 is not supposed to include file buffers within the program disk image. In assessing these program image sizes for running in limited memory space it should be kept in mind that for programs not linked to the runtime module the disk image will be considerably smaller than the memory image, since in that case the runtime module will be separately loaded into memory in addition to the program disk image.

Utilities

The Microsoft COBOL package includes both general program development utilities plus some others designed especially for the COBOL -80.

The general program development utilities include:

M80
Microsoft's 8080/Z80 macro assembler

L80
The linking loader mentioned before, which can be used also with M80 and FORTRAN-80

LD80
The *disk linker* version of L80

LIB80
An object library maintenance utility designed to be used with type REL, relocatable object files

CREF80
A utility for generating symbol table cross references

The special utilities are:

SEQCVT.COM
For converting MS-COBOL data files from SEQUENTIAL format to to LINE SEQUENTIAL

CVISAM.COM
For converting indexed files, created before version 4.6, to new format

REBUILD.COM
For recovering damaged indexed files

RECOVR.COB
A skeleton MS-COBOL program for reading and rewriting indexed files (thus eliminating some problems that may remain in files created with previous versions of COBOL-80)

Additional files included are:

SQUARO.COB
A square root program written in COBOL, useful in checking out the operation of the compiler

CRTEST.COB
A COBOL program that uses the CRT driver, allowing the user to check out the CRT driver module module

COPCOB.SUB
A submit file to assist in making copies of the distribution disk

Enhancements

Version 4.65 offers both some enhancements and some problem corrections from previous versions, some of which have not been covered so far:

As alluded to previously, file buffers for sequential, line sequential and relative type files are now allocated outside the program image, thereby allowing for a reduction in program disk image size. Additionally, relative access files are made contiguous by filling space between records with zeroes, thus allowing such files to be copied using the CP/M PIP utility.

The problem with loss of data from files left open while CHAINing to another program has been fixed.

Some help is now provided in the case of an out-of-space condition when writing data to disk files.

A problem in performing REWRITEs with indexed files containing variable length records has been resolved.

Additionally, problems with evaluation of subscripted variables, with the exponentiation algorithm, with the SORT facility and with correct handling of cursor and screen field attributes during console I/O have been resolved.

Summary

COBOL-80 is a fairly sophisticated implementation for 8-bit micros, and a small business user would certainly appreciate many of its advanced features, particularly the debugger, the screen controls, and the file-handling facilities. A more demanding application, however, will probably miss the multi-key file access features of the full ANSI COBOL.

Microsoft has been marketing its COBOL-80 for several years, and the system seems to still be in a state of evolution and improvement. Experience indicates that at this point not all of the bugs have been found and corrected, and certainly not all the enhancements have been devised and implemented. ■

Software Notes

Tips and Techniques

For A Flip-Flop When The Cycle Is A Power Of Two

by John Coggeshall

Avoid division. It's as tiresome for a computer as for a person. The same goes for the MOD function, which uses division. Here's a way to implement a flip-flop when the desired cycle is a power of two.

Any positive integer modulo (b±n) is the rightmost n digits of the integer as represented in base b. In particular, an integer modulo (2±n) is simply the rightmost n bits of the binary (the computer's) representation. Most languages have logical operators, providing an easy way to isolate those digits:

```
Mod2.n% = Integer% AND
[2±n−1],
```

where the programmer inserts the proper constant in place of the bracket expression.

For example,

```
Year.from.a.LeapYear% = Year%
AND 3 (used by Steven Fisher in
Lifelines, March 83, p.11); or
Two.Way.FlipFlop% = Index%
AND 1 (to accept Robert Van
Natta's invitation, ibid., p.9); or
IF Index% AND 1FH THEN [ not a
multiple of 32 ]
```

And so forth. ■

(continued from page 17)

Third, get the family involved in the business. Don't become a computer guru, hiding away in the closet. Get your spouse to help with answering the phone, typing correspondence, making bank deposits, etc. Get the children to fold mailers and stuff envelopes. Get everyone involved with the use of the computer. You'd be surprised how many family members believe that the only function of the computer is to play games! Show them how it can be used to expand your and their horizons and you will do them and you a favor!

Fourth, be persistent. This one item should be at the top of the list. A great many people feel that they should be able to form a business; accumulate all the trappings of being in business; and then sit back and wait for the money to come rolling in. The computer doesn't add any money magic to your business. Only you can do that. Take things slow and sure and expect some setbacks. There are fortunes being made with the use of computers, but it is the persistence of the people behind the ideas and equipment that make those fortunes possible. The ability to stay on a course of endeavor in the face of setbacks and good times makes profitable ventures!

J. Norman Goode is the Editor and Publisher of Micro Moonlighter Newsletter, *a publication for those wishing to use their computer for part- or full-time income. Subscriptions are $30 per year — samples are available for $3. He can be reached at 4121 Buckthorn Court, Lewisville, TX 75028.*

Letters to the Editor

To the Editor:

I wanted a patch so as to be able to keep my SUPERBRAIN disks compatible with my KayPro running UNIFORM. My SUPERBRAIN has 40 tracks, single side, double density. In looking around in Micro Solution's UNIFORM with DDT, I discovered that the newly acquired version I just updated from my dealer was a bit different from UNIFORM's earlier versions of SETDISK and INITDISK. I have found the addresses to patch to change the SUPERBRAIN format from 35 tracks with 165K to 40 tracks with 190K and I've included them for your readers.

	Address	Change From	Change To
SETDISK, Version 1.01	1022	51	5E
SETDISK, Version 1.02	109F	51	5E
INITDISK, Version 1.0	0E1C	23	28
INITDISK, Version 1.02	0E45	23	28

UNIFORM allows the KayPro II to read and write to 13 additional disk formats and is produced by:

Micro Solutions, Inc.
125 S. Fourth St.
De Kalb, IL 60115
(815) 756-3421

Bob Hickey
Eagle River

To the Editor:

In the interest of sharing knowledge on keeping up with new releases of CP/M, I am enclosing a small patch for WordStar version 3.01P which may help other users who are trying to make it run under ALS's CP/M version 3.0 on the Apple II.

Location	New Contents (in HEX)
0101	08 (Only patch in WSMAKER that is unique to CP/M 3.0)
0102	2D
049B	C3 (Makes backspace operate like delete, to delete character to left.)
049C	67

The patches at 049B and 049C were made easy to figure out because of Robert VanNatta's article in the April 1982 issue of *Lifelines*. I hope other users, trying to get software which ran under CP/M 2.2 to now run under version 3.0, from ALS, will publish their results. Apparently the new release is not quite as upward compatible as it could be.

As people find patches for programs like MBASIC, Z-Term, and Infostar, I hope *Lifelines* might publish their findings. And if anyone learns how to eliminate the redundancy of having CP/M use up 21K bytes on a cold-boot disk in addition to the system tracks, I'm sure quite a few people would be eager to learn the secret.

John E. Stith Colorado Springs, Colorado

Feature

CB80/CB86 Graphics

by Steven Fisher

Having recently purchased a TeleVideo 912 terminal that emulates a Tektronix 4012 graphics terminal, I developed a set of hardware-specific control functions with which I could experiment (that means "so I could play with my new toy"). The hardware imitating the 4012 is DataType's Mark III; it is very similar to the Retrographics RG512 from Digital Engineering.

The hardware

Tektronix has become the *de facto* standard graphics terminal, offering 1024 horizontal by 780 vertical directly addressable data points. Large libraries of graphics software have been developed using Tektronix' *Plot-10* and ISSCO's *DISSPLA;* these languages "talk" to the 4010 terminal series.

An after-market grew out of the combination of Tektronix' widespread use and its high price. True, you get your money's worth in high resolution and fast graphics generation, but many of us are not comfortable spending over $3,500 for a terminal. The DataType graphics adaptor lists for $695; the Mark II fits TeleVideo 925 and 950 terminals, while the Mark III works with the TVI 910, 912 and 920. DataType also markets a detached graphics cursor keypad, the Dot Director, for $195.

The DataType Mark II/Mark III and Dot Director graphics modifications for TeleVideo terminals are available from many TeleVideo distributors, or direct from the manufacturer. They are located at 2615 Miller Avenue, Mountain View, CA 94040. Their telephone number is (415) 949-1053.

Considering the deep discounts available on the popular TeleVideo units, you should be able to acquire a new graphics terminal for well under $1,500. If you already own a TeleVideo, you can have your local service organization upgrade it with a graphics board. Normal terminal operation is unaffected by the Mark II or Mark III.

The Tektronix video monitor provides higher resolution than the TeleVideo unit, so the 4012's 1024 by 780 screen is "mapped" via firmware into the DataType's 512 by 250 display. This scaling technique means that Tektronix software can run without being changed; a circle still looks round on the DataType. This avoids bothersome "aspect ratio" calculations necessary on other graphics units, such as the IBM-PC or Apple-II.

A graphics-modified TeleVideo operates in four modes: normal text, graphics text, point-plot, and vector-plot. In normal mode, the graphics functions are dormant. The graphics-text mode displays as many as 35 lines of text, each containing up to 73 characters. The point-plot mode draws a dot wherever the graphics cursor is positioned. The vector-plot mode generates a line between the current graphics cursor location and its prior location, unless the current location was the first one set since entering the vector-plot mode.

You can invoke the Hardware Graphics Cursor (HGC), a blinking crosshair. Pressing the directional arrow keys moves the HGC. If you have installed the Dot Director keypad option, you can move the crosshair diagonally by pressing two arrows simultaneously. When you press the keypad's <HIT> key, or any keyboard key, the graphics board transmits the depressed key followed by a four-character row and column location. The row and column coordinates are given in the Tektronix scale, and are therefore accurate within five rows (Y) or two columns (X).

The graphics board responds to a status request with a one-character status followed by the same four-character graphics cursor location. Since there are delays associated with some of the functions performed by the graphics board, it is good practice to have application software wait for status responses occasionally.

The graphics background (screen) can be either black or white. The plotting color is controlled separately, so you can erase by redrawing in the background color. The contents of the physical graphics screen can be read by a host processor one-half a scan line at a time; the video memory is represented as a stream of ASCII-HEX digits. Whenever a zero would be generated, it is replaced by "#n," where "n" is a binary count of the number of zero digits.

The software

The CB80/CB86 CBASIC compiler from Digital Research allows multiple-line user-defined functions. These functions may or may not require input data (parameters), and they may optionally return a value to whatever code segment invokes them. It is possible to write CB80/CB86 code that resembles PL/I, Pascal, or a verbose flavor of C. CB80/CB86 is a professional tool, so it allows separate compilation of code modules and the building of subroutine libraries.

All the hardware-specific functions have been defined within the file DATATYPE.BAS (Figure 1). They adhere to the credo that subroutines have a single entry point and a single exit, that modules be loosely bound with a minimum of external data (parameters), and that subroutines perform a single function.

They are written to use standard facilities of the CB80/CB86 language. One function (READLN) fetches graphics feedback without echoing to the screen; this function echoes under CP/M 1.4 and therefore fouls up the graphics display. Between floating-point calculation delays and judicious status checking, data overrun is avoided even at 9600 baud. Two of the routines (MIN%, MAX%) are "local," used only within the file.

The other routines are "public" and are accessible by external program segments.

Before a program segment can use the DATATYPE.BAS routines, it must define the subroutine names, their parameters, and what result value they may generate. This linkage data is defined within DATATYPE.EQU (Figure 2). Programmer/analysts spend a lot of effort finding the easy way to do things; they strive to "work smarter, not harder." Rather than retyping this information into every graphics program, CB80/CB86 authors can INCLUDE it into their source program when compiling.

GRDEMO.BAS (Figure 3) is an interactive graphics demonstration program. Since its compilation and usage instructions are contained within the listing, they shall not be repeated here.

The program lets you exercise every hardware/software option except the screen dump, and it also serves as an addictive experimental easel.

The final program, HOLE.BAS (Figure 4), displays the speed of program-generated line drawing. It is a modified version of the Apple II routine by Douglas Arnott that appeared in the December 1982 *Byte*.

The software accompanying this article is available in both source and compiled form on diskette from the author. Diskettes are shipped via U.S. Mail after receipt of $25 prepaid by either check or money order. California residents must include sales tax or a duly completed resale card. No UPS or COD. Allow four weeks for delivery. Available diskette formats are:

STD 8	8″ Standard CP/M (soft-sectored SS-SD)
DDNS	5.25″ NorthStar Lifeboat CP/M 2 (10-sector 48TPI SS-DD)
QDEP	5.25″ Epic Episode CP/M 2 (soft-sectored 96TPI DS-QD)

All diskettes will be 8″ unless otherwise requested. Address correspondence to Controlled Information Environments, P.O. Box 457, La Mesa CA 92041.

Future plans

The idea behind getting a graphics terminal was to evaluate and use the Graphics System Extension (GSX) software now offered by Digital Research. It provides an International Standards Organization (ISO), Virtual Device Interface (VDI) through a Basic Disk Operating System (BDOS) call. This keeps the application program independent from the graphics device. I had to know how to work my hardware before implementing its graphics interface.

My Graphics Input Output System (GIOS) is now over 28 kilobytes of structured 8080 assember source code, and it seems about half done. Expect to see a lot more on that later (pun intended).

Meanwhile, adapting these routines to your own graphics hardware is easier. So is using them as they are with Tektronix, Retrographics, or DataType hardware.

Figure 1. DATATYPE.BAS: CB80/CB86 Graphics Driver Subroutines

```
rem—    CB80 routines for DataType Mark II/III Graphics Mod for
rem—    TeleVideo
rem—    Copyright (C) August 19, 1983 by Steven Fisher CDP
rem—    Version 1.9 as of 9/04/83

rem—    The DataType Graphics Mods make TVI 912/920/925/950
rem—    emulate a Tektronix 4012 terminal, mapping 1024 by 780
rem—    points into the physical screen size of 512 by 250 points.
rem—    This hardware add-on closely resembles the Retro-
rem—    graphics RG512 by Digital Engineering.

rem—    To include these routines at link time, use the
rem—    command:
rem—      LK80 program[Q],DATATYPE,CB80.IRL

rem—    return greater of two integers

def     max%(first%,second%)
        if (first% gt second%) then \
          max% = first% \
        else \
          max% = second%
        return

fend

rem—    return lesser of two integers

def     min%(first%,second%)
        if (first% lt second%) then \
          min% = first% \
        else \
          min% = second%
        return

fend

rem—    erase the normal terminal screen

def     erase public
        print chr$(1bh) + "*";       rem clear TVI screen
        return

fend

rem—    read line of graphics reply without screen echo
rem—    will not work without echo for CP/M 1.4

def     rdline public
        string rdline

        temp$ = “ ”
        chr% = 0
        while (chr% ne 0dh)
        chr% = inkey
        if (chr% ne 0dh) then \
          temp$ = temp$ + chr$(chr%)
        wend
        rdline = temp$
        return
        fend

rem—    invoke the hardware graphics cursor and accept the
rem—    location
rem—    return termination key and Tektronix coordinates

def     cursor public
        string cursor

        temp$ = “ ”
        while (len(temp$) lt 4)
          print chr$(1bh) + chr$(1ah);
          temp$ = rdline
        wend
        cursor = right$(“ ” + temp$,5)
        return

fend

rem—    fetch the graphics status, waiting for a valid reply length
```

```
def       status public
          string status
          temp$ = " "
          while len(temp$) lt 5
             print chr$(00h) + chr$(00h) + \
                chr$(00h) + chr$(00h) + chr$(00h);
             print chr$(00h) + chr$(00h) + \
                chr$(00h) + chr$(00h) + chr$(00h);
             print chr$(1bh) + chr$(05h);          rem— ask for
             temp$ = rdline                        rem— status after
          wend                                     rem— delay
          status = right$(temp$,5)
          return
fend

rem—      read specified segment of physical graphics display
rem—      segments go from upper left corner of display, 2 to a line
def       grdump(scanline%) public
          string grdump

          if ((scanline% lt 0) or (scanline% gt 499)) then \
             temp$ = "=" + string$(64,'0') + ";" \   give empty line
          else \
             temp$ = " " :\
             msb% = shift(scanline%,5) :\
             lsb% = scanline% and 1fh
          while ((left$(temp$,1) ne "=") and (right$(temp$,1) ne ";"))
             print chr$(1bh) + chr$(11h) + chr$(msb%) + chr$(lsb%);
             temp$ = rdline
          wend
          temp% = match("\#",temp$,1)
          while (temp%)
             front$ = left$(temp$,temp%-1)
             back$ = right$(temp$,len(temp$) + 2-temp%)
             pad% = 1fh and asc(mid$(temp$,temp% + 1,1))
             temp$ = front$ + string$(pad%,'0') + back$
             temp% = match('\#',temp$,temp%)
          wend
          grdump = mid$(temp$,2,len(temp$) - 1)
          return
fend

rem—      turn on normal terminal option

def       alpha public
          print chr$(1ch) + chr$(1fh) + chr$(18h);
          return
fend

rem—      turn on graphics text option

def       graph public
          print chr$(1ch) + chr$(1fh);
          return
fend

rem—      start drawing vectors, next point is invisible until second
rem—      point

def       vector public
          print chr$(1dh);
          return
          fend

rem—      start drawing points

def       point public
          print chr$(1ch);
          return
fend

rem—      position graphics cursor to Tektronix location

def       plot(x%,y%) public
          x.msb% = 20h or shift(x%,5)
          x.lsb% = 40h or (1fh and x%)
          y.msb% = 20h or shift(y%,5)
          y.lsb% = 60h or (1fh and y%)
          print chr$(y.msb%) + chr$(y.lsb%) + \
             chr$(x.msb%) + chr$(x.lsb%);
          return
fend

rem—      clear graphics to dark screen and use light data points

def       black public
          integer black

          print chr$(1bh) + chr$(0ch);
          temp$ = status
          black = - 1
          return
fend

rem—      clear graphics to light screen and use dark data points

def       white public
          integer white

          print chr$(1bh) + chr$(0ch) + chr$(1bh) + chr$(19h);
          temp$ = status
          white = 0
          return
fend

rem—      use dark data points in future

def       dark public
          integer dark

          print chr$(1bh) + chr$(7fh);
          dark = 0
          return
fend

rem—      use light data points in future

def       light public
          integer light

          print chr$(1bh) + chr$(61h);
          light = - 1
          return
fend

rem—      reverse existing screen and data point colors

def       invert(color%) public
          integer invert

          print chr$(1bh) + chr$(12h);
          temp$ = status
          invert = not color%
          return
fend

rem—      erase prior vector or pair of points,
rem—      leaving earliest point active

def       undraw(old.x%,old.y%,new.x%,new.y%,color%) public
          if (color%) then \
             temp% = dark \
          else \
             temp% = light
          call plot(new.x%,new.y%)
          call plot(old.x%,old.y%)
          if (color%) then \
             temp% = light \
          else \
             temp% = dark
          temp$ = status
          return
```

```
fend

rem—    determine if status indicates there is no hardcopy device

def     noprnt(status$) public
        integer noprnt

        temp% = asc(left$(status$,1))
        if (temp% and 16) then \
           noprnt = -1 \
        else \
           noprnt = 0
        return
fend

rem—    determine if status indicates no graphics text is accepted
rem—    no text is accepted when in the point or vector mode

def     notext(status$) public
        integer notext

        temp% = asc(left$(status$,1))
        if (temp% and 4) then \
           notext = 0 \
        else \
           notext = -1
        return
fend

rem—    determine if status indicates the Tektronix
rem—    Plot-10 margin is set

def     margin(status$) public
        integer margin

        temp% = asc(left$(status$,1))
        if (temp% and 2) then \
           margin = -1 \
        else \
           margin = 0
        return
fend

rem—    determine Tektronix graphics X coordinate
rem—    from device reply

def     getx(coord$) public
        integer getx

        temp$ = right$("    " + coord$, 4)
        getx = (32 * (31 and asc(mid$(temp$,1,1)))) + \
           31 and asc(mid$(temp$,2,1)))
        return
fend

rem—    determine Tektronix graphics Y coordinate
rem—    from device reply

def     gety(coord$) public
        integer gety

        temp$ = right$("    " + coord$, 4)
        gety = (32 * (31 and asc(mid$(temp$,3,1)))) + \
              (31 and asc(mid$(temp$,4,1)))
        return
fend

rem—    draw a box, given the opposite corners

def     dobox(old.x%,old.y%,new.x%,new.y%) public
        call vector
        call plot(new.x%,new.y%)
        call plot(new.x%,old.y%)
        call plot(old.x%,old.y%)
        call plot(old.x%,new.y%)
        call plot(new.x%,new.y%)
        temp$ = status
        return
fend
rem—    draw a filled block, given the opposite corners
rem—    use the longest vectors so the terminal does the work

def     dobloc(old.x%,old.y%,new.x%,new.y%) public
        delta.x% = new.x% - old.x%
        delta.y% = new.y% - old.y%
        if ((delta.x% ne 0) and (delta.y% ne 0)) then \
           if (delta.x% lt delta.y%) then \
              for x% = old.x% to new.x% step sgn(delta.x%) :\
                 call vector :\
                 call plot(x%,old.y%) :\
                 call plot(x%,new.y%) :\
                 temp$ = status :\        rem— wait until done
              next x% \
           else \
              for y% = old.y% to new.y% step sgn(delta.y%) :\
                 call vector :\
                 call plot(old.x%,y%) :\
                 call plot(new.x%,y%) :\
                 temp$ = status :\        rem— wait until done
              next y%
        call vector
        call plot(new.x%,new.y%)
        return
fend

rem—    draw a circle, given the center and a point
rem—    on the circumference
rem—    circumference-plotting algorithm adapted from
rem—    Daniel L. Lee, May 1983 DDJ

def     docirc(base.x%,base.y%,edge.x%,edge.y%) public
        delta.x% = abs(base.x% - edge.x%)
        delta.y% = abs(base.y% - edge.y%)
        hypotenuse = (float(delta.x%) ↑ 2) + (float(delta.y%) ↑ 2)
        tangent = sqr(hypotenuse / 2.0)
        limit% = int%(0.5 + tangent)            rem—45 degrees
        call point
        for offset% = limit% to 0 step -1
        tangent = tangent + (float(offset%)/tangent)
        temp% = int%(0.5 + tangent)
           if ((base.x%-offset% ge 0) and (base.y%-temp% ge 0)) \
           then \
              call plot(base.x%-offset%,base.y%-temp%)
           if ((base.x%-offset% ge 0) and (base.y% + temp% le \
           779)) then \
              call plot(base.x%-offset%,base.y% + temp%)
           if ((base.x% + offset% le 1023) and \
           (base.y%-temp% ge 0)) then \
              call plot(base.x% + offset%,base.y%-temp%)
           if ((base.x% + offset% le 1023) and (base.y% + temp% \
           le 779)) then \
            call plot(base.x% + offset%,base.y% + temp%)
           if ((base.x%-temp% ge 0) and (base.y%-offset% ge 0)) \
           then \
              call plot(base.x%-temp%,base.y%-offset%)
           if ((base.x%-temp% ge 0) and (base.y% + offset% le \
           779)) then \
              call plot(base.x%-temp%,base.y% + offset%)
           if ((base.x% + temp% le 1023) and (base.y%-offset% \
           ge 0)) then \
              call plot(base.x% + temp%,base.y%-offset%)
           if ((base.x% + temp% le 1023) and (base.y% + offset% \
           le 779)) then \
              call plot(base.x% + temp%,base.y% + offset%)
        next offset%
        temp$ = status
        call vector
```

►

```
          call plot(edge.x%,edge.y%)
          return
fend

rem—      draw a filled circle, given the center and a point
rem—      on the circumference
rem—      circumference-plotting algorithm adapted from
rem—      Daniel L. Lee, May 1983 DDJ

def       dodisk(base.x%,base.y%,edge.x%,edge.y%) public
          delta.x% = abs(base.x% - edge.x%)
          delta.y% = abs(base.y% - edge.y%)
          hypotenuse = (float(delta.x%) ↑ 2) + (float(delta.y%) ↑ 2)
          tangent = sqr(hypotenuse / 2.0)
          limit% = int%(0.5 + tangent)          rem—45 degrees
          for offset% = limit% to 0 step - 1
            tangent = tangent + (float(offset%)/tangent)
            temp% = int%(0.5 + tangent)
            temp$ = status
            call vector
            call plot(max%(base.x%-offset%,0), \
              max%(base.y%-temp%,0))
            call plot(max%(base.x%-offset%,0), \
              min%(base.y% + temp%,779))
            call vector
            call plot(min%(base.x% + offset%,1023), \
              max%(base.y%-temp%,0))
            call plot(min%(base.x% + offset%,1023), \
              min%(base.y% + temp%,779))
            call vector
            call plot(max%(base.x%-temp%,0), \
              max%(base.y%-offset%,0))
            call plot(max%(base.x%-temp%,0), \
              min%(base.y% + offset%,779))
            call vector
            call plot(min%(base.x% + temp%,1023), \
              max%(base.y%-offset%,0))
            call plot(min%(base.x% + temp%,1023), \
              min%(base.y% + offset%,779))
          next offset%
          temp$ = status
          call vector
          call plot(edge.x%,edge.y%)
          return
fend

rem—      end of source
```

Figure 2. DATATYPE.EQU: CB80 Graphics Equates

```
rem—      CB80 equates for DataType MarkII/MarkIII Graphics Mod
rem—      for TeleVideo
rem—      Copyright (C) August 19, 1983 by Steven Fisher CDP
rem—      Version 1.7 as of 8/23/83

%nolist
rem—      The DataType Graphics Mods make TVI 912/920/925/950
rem—      emulate a Tektronix 4012 terminal, mapping 1024 by 780
rem—      points into the physical screen size of 512 by 250 points.
rem—      This hardware add-on closely resembles the Retrographics
rem—      RG512 by Digital Engineering.

rem—      To reference the routines within the program,
rem—      use the directive:
rem—                %INCLUDE DATATYPE.EQU

rem—      To include the routines at link time, use the command:
rem—      LK80 program [Q],DATATYPE,CB80.IRL

rem—      erase the normal terminal screen

def       erase external
          fend

rem—      read line of graphics reply without screen echo

def       rdline external
          string rdline
fend

rem—      invoke the hardware graphics cursor and accept the
rem—      location
rem—      return termination key and Tektronix coordinates

def       cursor external
          string cursor
fend

rem—      fetch the graphics status, waiting for a valid reply length

def       status external
          string status
fend

rem—      turn on normal terminal option

def       alpha external
fend

rem—      turn on graphics text option

def       graph external
fend

rem—      start drawing vectors, next point is invisible until
rem—      second point

def       vector external
fend

rem—      start drawing points

def       point external
fend

rem—      position graphics cursor to Tektronix location

def       plot (x%,y%) external
fend

rem—      clear graphics to dark screen and use light data points

def       black external
          integer black
fend

rem—      clear graphics to light screen and use dark data points

def       white external
          integer white
fend

rem—      use dark data points in future

def       dark external
          integer dark
fend

rem—      use light data points in future

def       light external
          integer light
fend

rem—      reverse existing screen and data point colors

def       invert(color%) external
          integer invert
fend

rem—      erase prior vector or pair of points,
rem—      leaving earliest point active

def       undraw(old.x%,old.y%,new.x%,new.y%,color%) external
fend

rem—      determine if status indicates there is no hardcopy device

def       noprnt(status$) external
          integer noprnt
fend
```

```
rem—     determine if status indicates no graphics text is accepted
rem—     no text is accepted when in the point or vector mode

def      notext(status$) external
         integer notext
fend

rem—     determine if status indicates the Tektronix Plot-10
rem—     margin is set

def      margin(status$) external
         integer margin
fend

rem—     determine Tektronix graphics X coordinate
rem—     from device reply

def      getx(coord$) external
         integer getx
fend

rem—     determine Tektronix graphics Y coordinate
remz-    from device reply

def      gety(coord$) external
         integer gety
fend

rem—     draw a box, given the opposite corners

def      dobox(old.x%,old.y%,new.x%,new.y%) external
fend

rem—     draw a filled block, given the opposite corners
rem—     use the longest vectors so the terminal does the work

def      dobloc(old.x%,old.y%,new.x%,new.y%) external
fend

rem—     draw a circle, given the center and a point
rem—     on the circumference
rem—     draw from the four extreme x,y locations to the 45-degree
         points

def      docirc(base.x%,base.y%,edge.x%,edge.y%) external
fend

rem—     draw a filled circle, given the center and a point
rem—     on the circumference

def      dodisk(base.x%,base.y%,edge.x%,edge.y%) external
fend

%list
```

Figure 3. GRDEMO.BAS: CB80/CB86 Graphics Demonstration Program

```
rem—      GRaphics DEMOnstration program, version 1.8 as of
rem—      8/24/83
rem—      Copyright (C) August 19, 1983 by Steven Fisher CDP

rem—      This program serves as a test-bed for the DATATYPE
rem—      control routines.
rem—      DATATYPE is specific to the DataType Mark II/III TVI
rem—      Graphics Mod.

rem—      To compile: CB80  GRDEMO [B]
rem—      To link:    LK80  GRDEMO[Q],DATATYPE,CB80.IRL
rem—      To run:     GRDEMO

%include DATATYPE.EQU

rem—      use Dot Director Cursor Pad to draw vectors, points,
rem—      boxes, circles
rem—      pressing <ESC> exits, <DEL> erases, <HOME> starts
rem—      new vector

def       draw(color%)
          prior.x% = 512
          prior.y% = 390
          cross.x% = 512
          cross.y% = 390
          call vector rem—start with crosshairs at center
          call plot(cross.x%,cross.y%)
                        rem—point invisible until connected
          coord$ = cursor
                        rem—get operator feedback
          while (left$(coord$,1) ne chr$(1bh))
            option$ = ucase$(left$(coord$,1))
            if (option$ eq chr$(1eh)) then \   HOME
            call vector                        rem—start new vector
            if (option$ eq "E") then \
              if (color%) then \
                call black :\                  erase to black
                call vector :\                 restore point
                call plot(cross.x%, cross.y%) \
              else \
                call white :\                  erase to white
                call vector :\                 restore point
                call plot(cross.x%, cross.y%) \
            else \
            if (option$ eq "I") then \
              color% = invert(color%) :\
              call vector :\                   restore point
              call plot(cross.x%, cross.y%) \
            else \
            if (option$ eq chr$(7fh)) then \   DEL
            call undraw(prior.x%,prior.y%, \
              cross.x%,cross.y%,color%) \
            else \    rem—all other options affect location
            if (match("\"+option$,chr$(1eh)+"  .BCDFT",1)) then \
              prior.x% = cross.x% :\
              cross.x% = getx(coord$) :\
              prior.y% = cross.y% :\
              cross.y% = gety(coord$)
            if (option$ eq ".") then \
              call point :\
              call plot(cross.x%, cross.y%) :\
              call vector \                    start new vector
            else \
            if (option$ eq "B") then \
              call dobox(prior.x%,prior.y%,cross.x%,cross.y%) \
            else \
            if (option$ eq "C") then \
              call docirc(prior.x%,prior.y%,cross.x%,cross.y%) \
            else \
            if (option$ eq "D") then \
              call dodisk(prior.x%,prior.y%,cross.x%,cross.y%) \
            else \
            if (option$ eq "F") then \
              call dobloc(prior.x%,prior.y%,cross.x%,cross.y%) \
            else \
            if (option$ eq "T") then \
              call vector :\                   start new vector
              call plot(cross.x%, cross.y%) :\ locate text
              call graph :\                    enable text
              input ""; line temp$ :\          get & echo
              call vector :\                   restore point
              call plot(cross.x%, cross.y%) \
            else \
            if (option$ eq chr$(20h)) or \     HIT
            (option$ eq chr$(1eh)) then \      HOME
              call plot(cross.x%, cross.y%)
          coord$ = cursor
                        rem—get operator feedback
          wend
          return
fend
rem—      Graphics Test Program main logic (such as it is)
call erase              rem—clear normal screen
```

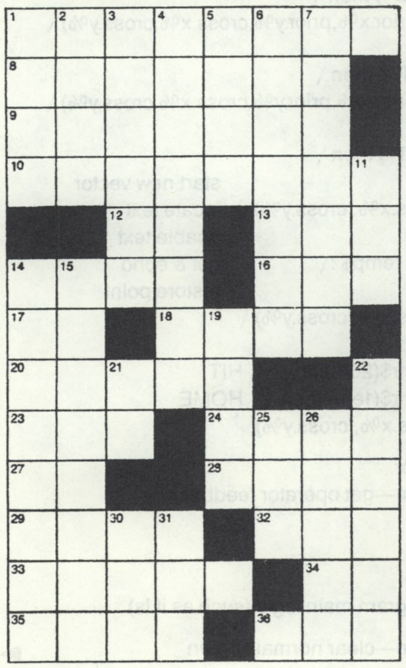
▶

```
print "GRDEMO Version 1.07 as of 8/23/83 by Steven Fisher CDP"
print "Copyright (c) 1983 Controlled Information Environments"
print "Portions Copyright 1982 Digital Research, Incorporated"
print
print "This program enables you to draw figures using
    the arrow keys,"
print "keyboard, and graphics pad of your
    TeleVideo/DataType terminal."
print
print "Pressing certain keys will cause the following:"
print " B         draws an open Box, using the current and prior"
print "             crosshair locations as opposite corners;"
print " C         draws an open Circle, with the current point on"
print "             the circumference and the prior point the center;"
print " D         draws a filled circle, a Disk;"
print " E         Erases the graphics screen;"
print " F         draws a Filled box;"
print " I         Inverts the video colors;
print " T         accepts Text until a CARRIAGE-RETURN;"
print " .         draws a single point;"
print " <HOME> marks the start of a new vector or figure;"
print " <HIT>     draws a vector from the current point to the"
print "             prior location. This is the same as SPACE;"
print " <DEL>     Deletes the previous vector or pair of points;"
print " <ESC>     ends this program."
print "                 Press any key to continue. . .";
temp% = inkey
call erase                      rem—clear normal screen
call vector                     rem—get into a graphics mode
color% = white                  rem—draw on white background
call vector                     rem—locate prompt text
call plot(0,0)                  rem—lower left corner
call graph                      rem—enable text display
print " B(ox) C(ircle) D(isk) E(rase) F(ill) I(nvert) T(ext)";
print " . HOME HIT DEL ESC";    rem—jog operator's memory
call draw(color%)               rem—draw until ESCAPE pressed
color% = black                  rem clear graphics to black
call alpha                      rem—return to normal terminal
        end
rem—    end of source
```

Figure 4. HOLE.BAS: CB80/CB86 Graphics Program

```
rem—      HOLE demonstration program, version 1.0
rem—      as of 8/25/83
rem—      recast from DEC 82 BYTE article (p 398-403) by
rem—      Douglas R Arnott

rem—      This program uses the DATATYPE graphics
rem—      control routines.
rem—      DATATYPE is specific to the DataType Mark II/III
rem—      Graphics Mod.

rem—      To compile:   CB80  HOLE [B]
rem—      To link:      LK80  HOLE[Q],DATATYPE,CB80.IRL
rem—      To run:       HOLE

%include DATATYPE.EQU

def       fnx%(degree,value) = 450 + int%(value * cos((degree *\
          6.28)/360.0))
def       fny%(degree,value) = 390 - int%(value * sin((degree * \
          6.28)/360.0))

rem—      Graphics Program main logic
          call erase          rem—clear normal screen
          call vector         rem—get into a graphics mode
          color% = black      rem draw on black background
          for z = 150.0 to 350.0 step 50.0
            for i = 0.0 to 360.0 step 20.0
              call vector
              call plot(650,390)
              call plot(fnx%(i,z),fny%(i,z))
              call plot(fnx%(i + 20.0,z),fny%(i + 20.0,z))
            next i
          next z
          call vector         rem—locate prompt text
          call plot(0,0)      rem—lower left corner
          call graph          rem—enable text display
          print " Press any key to continue . . .";
          temp% = inkey
          color% = black      rem—clear graphics to black
          call alpha          rem—return to normal terminal
          end

rem—      end of source
```

Crossword Puzzle by Crescent Varrone



Across

1. Early Hitchcock film.
8. Skill in public speaking, or a small chapel.
9. Greek letter, or an invocation to a metric unit.
10. Place for wrestling in ancient Greece.
12. Latin conjunction.
13. Egypt formally, formerly.
14. Point of convergence on a diagram or chart.
16. Direction.
17. Sus. Sawyer, e.g.
18. Newsman Hughes.
20. Eve was the first.
23. Tax break or Gershwin.
24. Cyclic Redundancy Check Character.
27. Senior circuit.
28. Modern Siamese.
29. Brazilian futeboler.
32. This iss.
33. CP/M __ Group.
34. Electricity before Steinmetz, or comic book publisher.
35. Russian Caesar.
36. Carbonic remains.

Down

1. Infinite or Chicago's.
2. La Douce.
3. Did not succeed.
4. And so forth.
5. Arthurian knight.
6. Sonorous or pompous.
7. *Anthem* writer.
11. Second person form of **to be**.
14. Johnny-come-lately, to a computer.
15. Water quality.
19. Roman Catholic sacr., Extreme __.
21. Follows B.A.
22. Usually precedes a gritch.
25. Density symbol.
26. Computer-aided circuit designs.
30. Grassy field.
31. Wander from the right path.

Software Notes

by Todd Katz

When working with microcomputers it doesn't take long to learn the importance of backing up text files frequently. When working with PMATE's text buffer, doing this is as easy as typing XE on the command line. XE tells PMATE to save the file of record which has been named by the XF command (see the manual if you are thoroughly confused).

But what if you are interested in saving a file that is not the file of record OR a file that is in one of PMATE's 10 buffers rather than the text buffer? Well, you can save a file using the XX and XO instructions. First you XXfilename the file and then XOfilename the file. This is necessary because PMATE will not allow you to XO over an already existing file name.

After doing this a few times I decided that a macro could make the file saving process at least as easy as the XE command and, at the same time, help in the identification of files.

The trick is to put the name of the file as a header line preceded by a semicolon. Semicolons, as you will recall, are treated as comment lines by PMATE, exactly like the . . commands in MicroPro's WordStar.

And after the file is named you can put in information like the date, version number of the program, etc.

But what if you try to use this macro, which I named the ";" macro, without a commented file name line at the top of the file? In such a case the program calls the "X" macro, which builds a four line "window" on the screen and prompts for the name of the file you want deleted, then saved. You type in the name and the macro prompts for the disk drive to which you want the file saved. The only choices currently are "A" and "B" drives but obviously others could be added quite easily. After naming the drive, the operation described above takes place — the program you named is deleted (XXed) from the disk (assuming it exists) and then the current buffer's file is saved to the appropriate file name and drive (XOd).

It's really much more difficult to describe than to use.

One last note: the "X" macro can be used as a window for a variety of macro tasks. In the listing we have used a high bit set character but an asterisk could as easily be substituted.

```
ua            ; go to the first line in the file
@t=';{        ; IF char. at cursor is a semi-colon
  mt          ; move one character forward and tag spot
    {         ; move cursor forward until a character
     m(@t<32)
              ; with an ASCII value below 32
  }           ; end of loop
  #           ; take characters between tag and cursor position
  b8c         ; and copy them to buffer 8
  xx↑A@8$     ; delete any file with the same name as in buf. 8
  xo↑A@8$     ; save the file in the text buffer to that name
  ua          ; go to the first line in the file
 0gOK$        ; flash message "OK"
 }{           ; ELSE
 .x           ; invoke macro "X"
 }
; end of macro 1

0L            ; go to beginning of line
80ff          ; set column width to 80 and turn off EOL markers
t             ; tag cursor position
4[13i]        ; insert four carriage returns
#;
15[129i]      ; insert 15 high-bit characters
L             ; move one line
3[            ; do the upcoming three times
  14qh        ; insert 14 spaces
  1L-m        ; go to the end of the current line
  129i        ; insert 1 high-bit character
  L           ; move one line
  ]           ; end iteration
0L            ; move to the beginning of line
15[129i]      ; insert 15 high-bit characters
13i           ; insert carriage return
-3L           ; move backward three lines
t             ; tag the spot
.iName the current file you want deleted then copied$  ; insert prompt
0L            ; go to the beginning of the line
t             ; tag cursor position
gA or B drive?$  ; prompt
0gdeleting and saving$  ; report current activity
@ki           ; insert character (A or B)
i:$           ; insert colon (:)
qr            ; redraw screen
L-m           ; go to end of line
#             ; exchange tag and cursor
b8c           ; copy contents to buffer 8
-2L           ; move back two lines
5k            ; delete five lines
xx↑A@8$       ; delete file named in buffer 8
xo↑A@8$       ; save current file under name in buffer 8
ua            ; go to top of file
gOK$          ; print "OK" message
;end of macro 2
```

Here is the answer to the September crossword puzzle.

1 S	2 O	3 F	4 T	5 W	6 A	I	7 R	8 E	■
9 L	A	R	I	A	T	■	10 A	X	11 E
12 I	S	O	L	D	E	■	13 D	P	T
14 P	I	N	E	D	■	15 T	I	E	■
■	16 S	T	R	E	■	17 E	X	C	18 E
19 C	■	■	■	20 N	21 A	K	■	22 T	M
23 E	24 V	25 O	26 P	■	27 L	S	28 D	■	■
29 N	E	B	U	30 L	A	■	31 D	32 A	33 B
34 T	R	E	L	L	I	35 S	■	36 L	O
37 R	U	S	E	■	■	38 I	39 N	F	O
40 E	M	E	R	I	L	■	41 C	A	T

Oops!

Robert Cahn discovered this correction in listing A.5 of the article GRAPHSUB (September 1983 issue) while running his program.

Listing A.5

```
VECTOR:PROC (X,Y)

DCL (X,Y,XC,YC) FLOAT;
DCL (IX,IY,FLAG,PLOT) FIXED;
DCL (XPREV,YPREV) EXTERNAL STATIC FLOAT;
DCL PVFLAG EXTERNAL STATIC FIXED;

DCL SCALE ENTRY (FLOAT,FLOAT,FIXED,FIXED,FIXED);
DCL VECTA ENTRY (FIXED,FIXED);
DCL CLIP ENTRY (FLOAT,FLOAT,FLOAT,FLOAT,FIXED);

CALL CLIP (X,Y,XC,YC,PLOT);

/* THIS IS WHERE THE WORK GETS DONE */

    IF (PLOT = 0) THEN DO;
    CALL SCALE(XC,YC,IX,IY,FLAG);
    CALL VECTA(IX,IY);
    END;

XPREV = X; YPREV = Y;

RETURN;

END VECTOR;
```

(continued from page 2)

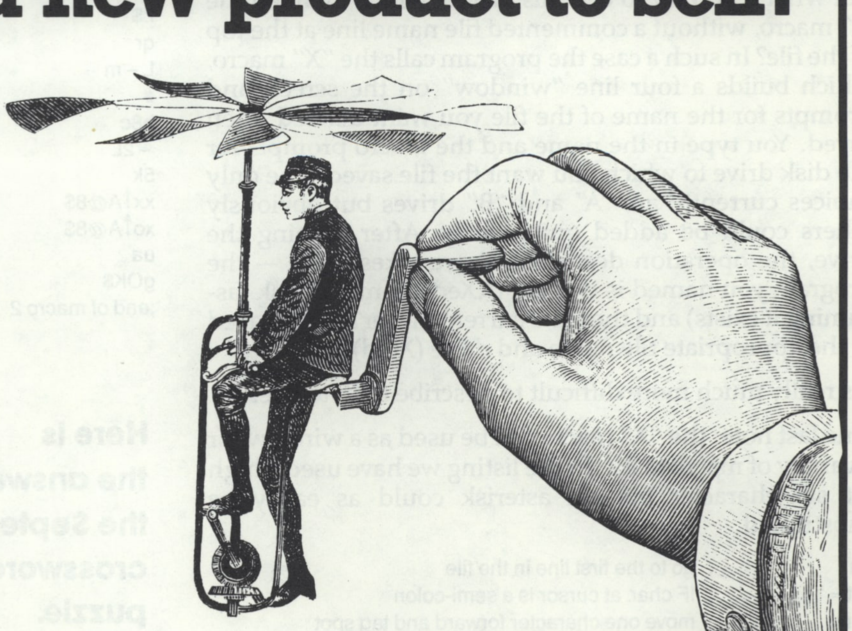
prove to be cost effective, reliable, easy-to-use and efficient or this will rapidly cease being the fastest growing field of technology in the world.

The technique of letting the end user do all of the testing must come to an abrupt end. This is as outmoded as the belief that software provided without support has a long term role to play in this market's future.

We cannot continue to provide alleged solutions in search of a problem any more than the industry intellectuals can continue to be those who have a better spreadsheet.

Let us have more matter and less art. . . . ■

Do you have a new product to sell?



Lifelines/The Software Magazine **is currently accepting advertising for future issues. If you have a product, book, or accessory and wish to reach thousands of microcomputer users, send for your free media kit and rate chart. Send all inquiries to: Will Lampe, Advertising Director, Lifelines Publishing Corp., 1651 Third Avenue, New York, NY 10028. Or call (212) 722-7100.**

Users Group Corner

Editors Note: We hope you will write in and give us information about your users group or computer club. Our Users Group Corner is designed to help you find computer clubs in your area or new clubs that your existing club can exchange information with.

Forth Interest Group
P.O. Box 1105
San Carlos, CA 94070
The FORTH Interest Group (FIG) is pleased to announce the recent formation of new chapters in Rochester, Syracuse, Denver and Dayton. A special graphics chapter (FIGgraph) has also formed in Santa Cruz, CA. They join the growing worldwide FORTH Interest Group community of over 40 chapters in 20 states and Australia, Canada, England, South Africa, Switzerland and Germany. Each chapter meets regularly to explore and share information about the FORTH computer language. Everyone is welcome — beginner to expert.

The "Forth Handy Reference Card" is now available free from the FORTH Interest Group. Functioning as a pocket programming aid, the card lists and describes the major commands of the FORTH computer language. This card serves as an excellent reference for all FORTH programmers. Commands are grouped by function for easy reference and use. These include: stack manipulations, number bases, arithmetic and logical, comparison, memory, control structures, terminal input-output, input-output formatting, disk handling, defining words, vocabularies, and system.

PicoNet
P.O. Box 391566
Mountain View, CA 94039-1566
Piconet, a CP/M Users' Group, has been formed as a non-profit organization for the purpose of assisting in the education and training of those involved with or interested in CP/M based systems. PicoNet also disseminates public domain software.

PicoNet has monthly meetings in Palo Alto on the campus of Stanford University in the auditorium of the linear accelerator (SLAC). The public is cordially invited.

Pico News is the official journal of PicoNet and is mailed to its members.

CPMUG
1651 Third Avenue
New York, NY 10028
The complete CPMUG™ catalog, which consists of nearly 100 volumes, is available for $10, prepaid to the United States, Canada, and Mexico; $15, prepaid to all other countries. (All checks must be in U.S. currency drawn on a U.S. bank.)

The following description is of Volume 56:

ORIGINAL ADVENTURE SOURCE CODE IMPLEMENTED FOR CP/M

NUMBER	SIZE	NAME
		CATALOG.056
		CONTENTS OF CP/M VOL.056
		ABSTRACT.056
		Implementation Notes
		SIG/M.LIB
		Submittal Form
		UGFORM.LIB
		Submittal Form
56.1	28K	ADINIT.COM
56.2	13K	ADVENSUB.FOR
56.3	5K	ADVENT.FOR
56.4	74K	ADVENTUR.DAT
56.5	29K	ADVINIT3.FOR
56.6	47K	ADVMAIN.FOR
56.7	5K	INSUB.FOR
56.8	5K	MAINSUB.FOR

Software in the library, obtainable exclusively on diskettes, is available for a prepaid media and handling charge, as follows:

FORMAT	DESTINATION
8″ IBM	U.S., Canada, Mexico—$13
8″ IBM	All other destinations—$17
North Star/Apple	U.S., Canada, Mexico—$18
North Star/Apple	All other destinations—$21
KAYPRO II	Same price as Apple II.

PLEASE CLEARLY SPECIFY THE FORMAT YOU WANT WITH YOUR ORDER

This payment covers the cost of the diskette(s), packaging, and postage. Domestic shipping is via UPS where a full street address is given; all other orders are via U.S. Postal Service. ■

Product Status Report

New Products

CHem-Cost
Silicon Forge
1 Softward Park
Bethany, CT 06525-3498
(203) 397-0051

This integrated product costing package, designed for chemical, paint and adhesive manufacturers, compounders and packagers, provides complete formula, batch and packaging cost information as well as finished goods pricing. Menus simplify operation, and user programability allows adaptation to meet special costing requirements. Built-in communication capabilities allow prompt remote technical service.

Requirements: IBM-PC, Apple II and IIe, Commodore 64 and most CP/M formats
Price: N/A

UNERA
Compu-Draw
1227 Goler House
Rochester, NY 14620
(716) 454-3188

This program is ideal for recovering accidentally erased files. File recovery is guaranteed if UNERA is used immediately after the erasure. However, even if other data has been written to the disk after the files were erased, UNERA recovers whatever

►

portions of the files it can. In such cases UNERA issues messages indicating whether the files were successfully recovered, partially recovered or have been permanently destroyed. By recognizing wild card characters in the name of the files to be recovered, UNERA supports recovery of multiple files in a single operation. Since it allows the user to change disks, it can conveniently be used in single-drive as well as multi-drive systems. UNERA adapts itself to the user's disk organization by requesting information from the CP/M CBIOS segment. It also makes several tests to ensure that only the required directory extents are recovered.

Requirements: CP/M (8080, 8085 or Z80 CPU), 16K
Price: $29

CRITICAL CONNECTIONS

USS Enterprises
6708 Landerwood Lane
San Jose, CA 95120
(408) 997-0264

This interface allows an Atari 400 or 800 to use the disk drives, printer and keyboard of any computer system running CP/M, as long as the system has a serial port at 19,200 baud. Some of the features include: automatic install for many systems, hardware that connects the CP/M serial port to the Atari disk/printer port which can be located up to 15 feet away, software on an 8″ single density disk which allows the CP/M disk drives, printer and keyboard to replace their regular Atari counterparts, and the ability to simulate, with the CP/M system, four Atari disk drives simultaneously.

Price: $175

PORTMAP

Single Source Solution
2699 Clayton Road
Concord, CA 94519
(415) 680-0202

This program allows the investor to analyze his investment portfolio, whether it contains stocks, bonds, moneymarket instruments, and/or other securities. PORTMAP's summary report analyzes cost basis, number of units, and expected income by account, type of security, month of payment, and taxability (Federal, State or non-taxable). The gain/loss report analyzes market values, gain/loss, and holding period by individual lot and type of security, margin account status is also reported. A special report containing all required data for Schedule D is provided. Any number of portfolios may be processed, and data may be selected by portfolio or individual accounts. All data entry is interactive and user-oriented.

Requirements: CP/M or MS-DOS
Price: $79.95

INSTAT

Single Source Solutions
(for address see above)

This program analyses datasets which contain extensive file handling and data manipulation features as well as data analysis programs. Three types of datafiles are supported: ASCII, System Data, and Matrix. Five data management programs are provided: DATIN — creates and enters data on an ASCII file; INPUT — creates a datafile or opens an existing file; RECODE — opens and reads data from a datafile and performs any desired data modifications (recording transformations, filtering of cases or filtering of variables); and REFILE — reconfigures a datafile. The following statistical procedures are supported: DESCRIBE — quick survey of variables with min, max, missing values, mean standard deviation; FREQ — frequency distribution; BREAK and BREAKOUT — displays the mean of the dependent variable for each of the independent variables; CROSS — cross-tabulations; PLOT — scatterplot of two variables; COR — correlation coefficients; MDCOR — missing data correlation program; CORREG — multiple regression analysis; FACTOR — factor analysis; DISCRIM — multiple discriminant function analysis; and FORECAST — variety of data smoothing options.

Requirements: CP/M
Price: $199.50

LEGAL TENDER

American CompuSoft
23113 Plaza Pointe Dr.
Laguna Hills, CA 92653
(714) 472-8186

This program enables a law firm of up to 25 lawyers to keep accurate records of client activity, billings, time charges, trust accounts, and calendars on one microcomputer. It is user-friendly and has been field proven for over two years in working law offices. LEGAL TENDER is menu-driven.

Requirements: CP/M, CP/M-86, MP/M, MS/DOS or PC/DOS
Price: N/A

BRAINSTORMER

Soft Path Systems
c/o Cheshire House
105 N. Adams
Eugene, OR 97402
(503) 342-3439

This program is designed to generate potential solutions to complex problems. The user is led through a series of steps to produce a structured representation of the problem that s/he is interested in solving. Then BRAINSTORMER guides the user through a process of examination and reconsideration of the structure by generating new ways of looking at the problem.

Requirements: CP/M, MBASIC, 2 drives, 48K
Price: $50

DISPATCH

Professional Publications Inc.
PO Box 199
San Carlos, CA 94070
(415) 593-9119

This order processing system for the mail order industry is designed to simplify mail order fulfillment. It utilizes a perforated multi-purpose form which incorporates two mailing labels, a packing list showing products and financial summary, a VISA/MasterCard sales draft and a refund check or credit memo. DISPATCH maintains inventory records, supports on-line customer service, and produces marketing reports about source codes, back orders, and inventory. Back orders are maintained automatically. It also contains a mailing list and accounting functions.

Requirements: CP/M, MS-DOS, 64K, 2 drives, 80 column screen
Price: N/A

LINK MODULE

Chang Labs
5300 Stevens Creek Boulevard
Suite 200
San Jose, CA 95129
(408) 246-8020

This program analyzes data files and loads them into a MicroPlan worksheet. With the Link Module, you can use MicroPlan to analyze data from mainframe files, timesharing services such as The Source, or files from other applications packages such as accounts receivable or database management software. When downloading files from mainframes or timesharing services, communications software is needed.The Link Module features posting and cross-tabulation commands. With these commands data can be posted either to individual or multiple rows or columns. In addition to reading files from other applications, MicroPlan tables can be read by applications such as database and word processing packages. With the Link Module, MicroPlan also reads DIF files.

Requirements: CP/M, MS DOS, MicroPlan, 64K, two drives.
Price: $295

Z80ASM and SLRNK

SLR Systems
1622 N. Main Street
Butler, PA 16001
(412) 282-0864

Z80ASM is a relocating macro assembler for the Z80 CPU utilizing Zilog mnemonics. The assembler features nestable macros, conditionals, and include files. Unique design allows one or two pass operations and will generate COM, HEX, Microsoft REL, or SLR Systems REL files with a throughput of over 6000 lines per minute. Z80ASM also generates a symbol table and cross-reference if requested. Both relocatable formats now support extended math operations on externals for bytes and words. SLR format is byte-oriented rather than a "bit-stream," supports global names of unlimited length, (Z80ASM and SLRNK currently allow up to 16 characters), and even allows external bit # for the BIT, SET, and RES instructions. SLRNK is the companion linking loader. It loads both Microsoft format and SLR format REL files, in any combination. Loading addresses may be specified for Program, Data, and even common blocks by name. Globals may be defined from the command line. SLRNK allows external bytes and expressions involving externals, and supports command-file operation.

Requirements: CP/M, Z80, 32K
Price: $169.95

ACNAP

BV Engineering
P.O. Box 3351
Riverside, CA 92519
(714) 781-0252

ACNAP (AC Network Analysis Program) is a general purpose electronic circuit analysis program which analyzes circuits consisting of resistors, capacitors, inductors, a voltage source, and controlled current sources. ACNAP will analyze the response of any linear network consisting of up to 21 nodes and 60 components. Every command is either menu driven or program prompted. Circuit topology data is saved to a named file or retrieved from a previously generated file for further analysis or editing. ACNAP is fast, calculating the response of a typical five-node circuit in 0.4 seconds. ACNAP works with component tolerances to provide Worst Case and Monte Carlo analysis. With a few short commands ACNAP calculates the minimum, maximum, mean, and three sigma points of a circuit's gain and phase response to any frequency input. Linear as well as logarithmic frequency sweeps are easily specified. The sensitivity of the gain/phase response to a particular component or components at a frequency or range of frequencies is similarly calculated from tolerance data. ACNAP automatically calculates any circuit's Noise Equivalency Band width. The user is prompted for all the information needed to accomplish these calculations. ACNAP will save the spectral data (magnitude/phase) to a file which, when used with BVE's "Signal Processing Program," enables the user to compute the response of his/her circuit to any time domain input waveform. This includes square waves, triangle waves, pulses, or any other user defined waveform.

Requirements: CP/M, TRSDOS, MSDOS
Price: $39.95

New Books

INTRODUCTION TO COMPUTER PROGRAMMING: A Problem-Solving Approach

Digital Press
12A Esquire Road
Billerica, MA 01862

INTRODUCTION TO COMPUTER PROGRAMMING, by John Hartling, Larry E. Druffel, and E. Jack Hilbing, introduces computer programming in terms of algorithmic problem solutions, starting in a very simple manner and building up through numerous examples to more sophisticated algorithms that provide insight into other areas of computer science. Unfortunately, most introductory programming courses require learning a new programming language. This book is language independent, emphasizing algorithm development.

In the earlier chapters of **INTRODUCTION TO COMPUTER PROGRAMMING** algorithms are expressed in flowcharts. In later chapters, after the student has become familiar and proficient with the flowchart technique and the development of the more basic algorithms, a shift is made to present algorithms through pseudocode. A large number of examples and problems are used to help teach each new concept. At the end of each chapter, over 100 supplementary problems are also included for further study.

Ordering Information: 1983,480 pages, hardbound. Order number: EY-00010-DP. Send all inquiries to above address.
Price: $24.00

THE HOTEL/MOTEL COMPUTER HANDBOOK

Computer Strategies
10218 Chimney Hill
Dallas, TX 75243
(214) 644-0222

Hotels and motels contemplating the purchase of a computer system can now turn to **THE HOTEL/MOTEL COMPUTER HANDBOOK** for help in selecting the right system and avoiding costly mistakes.

The handbook explains exactly what a hotel/motel computer system can ►

and cannot do, and how to justify the costs involved.

Checklists help the reader decide what features the system should have, and a fill-in-the-blanks request for proposal eliminates confusion and saves time in selecting from among the numerous hotel/motel systems available in today's marketplace. Negotiation of contracts and how to implement and manage a system is also covered. To help hotels and motels separate fact from fiction in vendor presentations, the book includes behind-the-scenes information from consultants and computer salesmen.

Price:$45.00 plus $2 shipping. Texas residents please add $2.25 sales tax. Overseas orders add $20.00 for airmail and remit in U.S. dollars.

New Versions

COGEN COBOL PROGRAM GENERATOR 6.0

Bytel Corp.
1730 Solano Avenue
Berkeley, CA 94707
(415) 527-1157

This upgraded package offers an integrated data dictionary that simplifies the entire user-interface, and provides an extensive array of new features, including permanent tailoring of generated code, full data validation, arithmetic calculations, conditional selection logic, and batch processing multiple files.

Users can add subroutines, write in comments, and change the naming conventions for field definitions. Programmers can also halt the generation of source code at any predetermined point enabling the insertion of additional command lines from the terminal. COGEN 6.0 allows programmers to specify up to nine conditions before data is accepted into any field, thereby providing comprehensive entry validation. A credit field, for example, might exclude negative values, while a check registry could limit entries to under $10,000. In addition, default values may be assigned to any data field to speed data entry by the operator. The new version allows programmers to create fields that are calculated from other fields. These, in turn, may be utilized within screens (displaying, for example, the accumulated item total), as well as on reports. COGEN 6.0 handles conditional selection logic, allowing the programmer or operator to specify which records are reported — the range of customers, for example, or the acceptable zip codes. Moreover, output may be directed to the screen (instead of the printer) enabling COGEN to function as an on-line query system.

The new version of COGEN also allows data to be updated in batch mode by generating batch control programs. Hence a change in credit rating, a raise in salary for all employees earning under $20,000, or a lower interest charge can be applied to many data files with no operator intervention.

COGEN's batch programs read a master file and up to 99 secondary files, directing the results to as many as 99 output files. The system can use complex criteria in deciding which records to process, and it performs extensive computations using intermediate work fields when necessary. Output may be further redefined with as many as nine conditional statements.

Requirements: UNIX
Price: $950

VERSAFORM 2

Applied Software Technology, Inc.
170 Knowles Drive
Los Gatos, CA 95030

Release 2 of VersaForm updates and expands the program's capabilities, including new features and modifications designed to improve the program's efficiency and increase its applications. The modifications include expanded capabilities on the mailing label printer; a 'copy by name' feature allowing data to be transferred from one form to another where field names are the same, providing the ability to redesign a form after data has been entered; and new report features including the ability to easily change report selection criteria at run time and a 'page break' option allowing grouped or subtotalled data to be printed on separate pages.

Price: $29.95

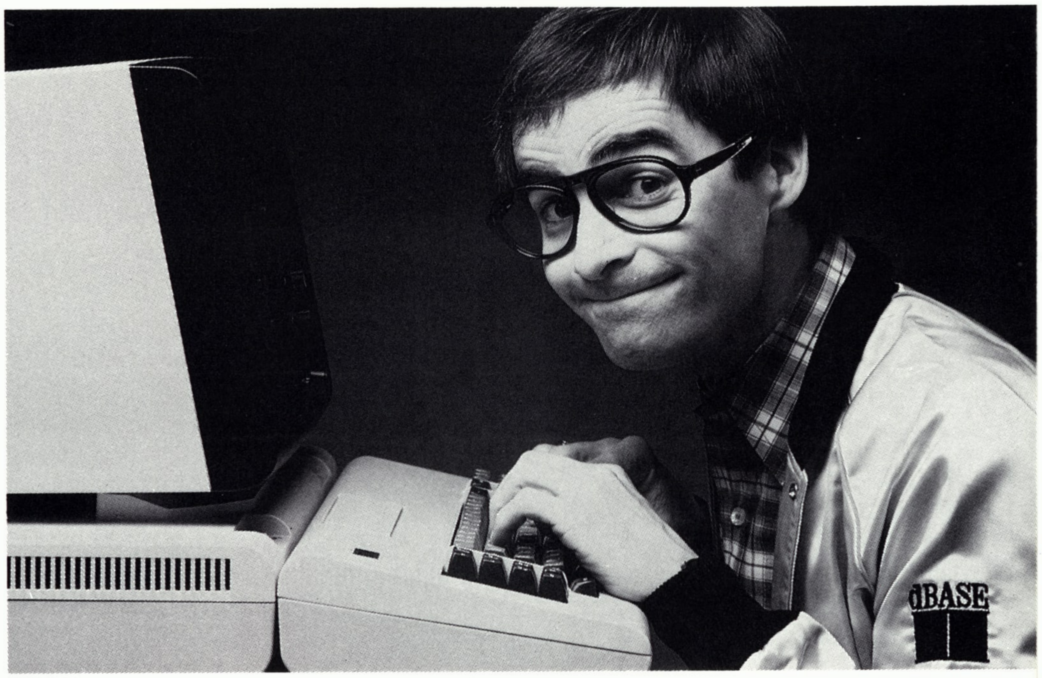
PLAN80 2.6

Business Planning Systems, Inc.
2 North State Street
Dover, DE 19901
(302) 674-5500

PLAN80 has the ability to consolidate any number of spreadsheets and transfer any number of values between sheets, automatically. It has an interactive spreadsheet mode and provides all of the common trig, math, financial and depreciation functions including ACRS, NPV (Net Present Value), IRR (Internal Rate of Return), plus AMORTization and SPREAD (for aging). It uses user-assigned names like SALES-COSTS = MARGIN for rows and columns. Models can be created very quickly because the model statements are entered with any familiar editor or word processor. Features include IF-THEN-ELSE logic, a screen graphics mode and fast data entry functions which will "grow" or replicate values automatically. Version 2.6 allows the user to transfer and consolidate parts of many different spreadsheets to a master spreadsheet by matching or equating row or column names. Users can further manipulate the consolidated data by adding additional rules in master sheet. Users can then consolidate the master data into higher level worksheets as needed. Values can be transferred from rows to columns or vice versa. Values can be referenced with a column SHIFT forward or backward. A "DYNAMIC VALUES" statement allows automatic insertion of a variable such as &DATE& throughout the model. Version 2.6 has extensive report formatting capabilities including variable decimal places, dash/zero or blank for nil values, brackets for minus, under/overscoring and spacing, $ and % signs and suppression of nil value rows. Models can be automated so that users can change one value and recompute and print a thirty page model with a single command. The operator can be prompted to input dates, titles, names, and values while the model is being computed. This allows PLAN80 models to be designed for use by inexperienced personnel.

Requirements: CP/M80, CP/M86, MSDOS, 56K memory for 8-bit systems and 128K for 16-bit systems.
Price: $295

All you dBASE II™ hotshots are about to get what you deserve.



You've written all those slick dBASE II programs.

Business and personal programs. Scientific and educational applications. Packages for just about every conceivable information handling need.

And everybody who sees them loves them because they're so powerful, friendly and easy to use.

But that's just not good enough.

Uh-uh.

Because now you can get the gold and the glory that you really deserve.

Here's how.

We've just released our dBASE II RunTime™ application development module.

And it can turn you into an instant software publisher.

The RunTime module condenses and encodes your source files, protecting your special insights and techniques, so you can sell your code without giving the show away.

RunTime also protects your margins and improves your price position in the marketplace. If your client has dBASE II, all he needs is your encoded application. If not, all you need to install your application is the much less expensive RunTime module.

We'll tell the world.

With your license for the dBASE II RunTime module, we provide labels that identify your program as a dBASE II application, and you get the benefit of all the dBASE II marketing efforts.

We'll also provide additional "how to" information to get you off and running as a software publisher sooner.

And we'll make your products part of our Marketing Referral Service. Besides putting you on our referral hotline, we'll publish your program descriptions and contact information in *dBASE II Applied*, a directory now in computer stores world-wide.

Go for it.

But we can't do any of this until we hear from you.

For details, write RunTime Applications Development, Ashton-Tate, 10150 West Jefferson Boulevard, Culver City, CA 90230.

Or better yet, just call (213) 204-5570. And get what you deserve today.

dBASE™ II

ASHTON·TATE

Lifelines™/ The Software Magazine™
1651 Third Ave., New York, New York 10028

Second Class Postage Paid
At Smithtown, N.Y.

[redacted]