felines™

The Software Magazine™

$3.00 September 1983 Volume IV, No. 4 (ISSN 02479-2575, USPS 597-830)

GRAPHSUB

by Robert S. Cahn

Lifelines

The Software Magazine

September 1983 Volume IV, Number 4

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

Electromagnetic Wastelands And The Twilight Zone

Hardware manufacturers are beginning to realize that the market accessible to them is decreasing in size in the 16-bit arena. They are comforted by the knowledge that while the percentage of the market which they may capture is decreasing, the absolute numbers are in fact increasing due to the phenomenal growth of this market.

Apple is undoubtedly paying close attention to IBM with respect to the anticipated announcement regarding the IBM "Peanut." This machine has allegedly already been shown to major accounts and is expected to be announced perhaps as early as October.

Rumors suggest that this machine is a stripped-down PC based on the 8088 and supporting MS DOS. Apparently there will be little or no expansion capability provided, as there is not to be a bus structure *á la* the IBM-PC.

The C language is continuing to gain ascendancy in the microcomputer celestial hemisphere and we expect that this trend will continue. Lattice C is leading the race in the 16-bit marketplace and Lattice Corporation is reportedly preparing many new exciting features.

Those of you interested in an excellent text editor for 8- and 16-bit machines should be taking a careful look at PMATE. A number of issues of *The Software Magazine* have included features discussing the various aspect of this excellent product. Macro of the Month has served as continual reminder of this, the most powerful of text editors for micros. If you haven't checked this one out, do so....

Interest in UNIX continues to grow and those of you aspiring to authorship of applications packages should give serious consideration to writing them in Lattice C (using PMATE, of course). New C books seem to be appearing weekly and we will review them as they appear. Fortunately, C is remaining relatively "pure" and is implemented on virtually everything from the mighty Cray 1 down to the lowest of microcomputers. C has a number of important attributes and the interest in C is growing faster than anyone might have imagined. So get busy, order a copy of your favorite C compiler, some books, a good text editor and start studying. You'll be up and running in no time. There are also a number of C user groups that we will review in future articles which are rapidly developing a large base of useful C programs.

If you are trying to crack the "database barrier" take a look at "Everyman's Database Primer," by Robert A. Byers. This book is published by Ashton-Tate and is an excellent treatment of databases and database management. The text is self-contained and exactly what you have been looking for if you are interested in,or involved with DBASE II or other data base managers. The text is extremely lucid and well illustrated with many examples.

Another interesting text is a book by Tom DeMarco entitled, "Controlling Software Projects—Management, Measurement and Estimation." This book, published by Yourdon Press, New York, is an interesting text for anyone charged with the responsibility of managing software projects. Perhaps not in the class of "The Mythical Man-Month," it nonetheless treats the general subject of software project management in an enlightened way. This is a fairly technical treatment and probably of little interest to the casual observer.

Sadly, it appears that computerized bulletin boards have entered a state of arrested development. There is little new software currently being supported by most of them and they seem to remain in a supersaturated state in terms of being able to access them. It was to be expected that the few CBBSs would never be able to meet the needs of the masses. Still, it's sad that we have reached this state so quickly.

Installation and maintenance of these facilities is done on a no-charge basis and represents a considerable investment in time and dollars on the part of the Sysops. Perhaps these bulletin board systems are not unlike the Pony Express of old. They served us well for a brief period but their memory has far more significance than their actual contribution. A salute to those who served us all so well if only for a short time. "Out, out brief candle."

It's not difficult to understand why ham radio is suffering such lack of interest these days. One has only to listen to life as we know it on the two meter bands to recognize that the totality of all transmissions in this band probably has less value than the energy required to propagate such nonsense.

If only the FCC would recognize the contribution that micros offer this important area of technology. The computer bulletin boards of the future could broadcast continuously to all those within range and offer a wealth of public domain software to one and all. Furthermore this would breathe new life into the interfacing of micros with ham equipment and perhaps even breed a hybrid of individuals skilled in micro and ham radio technology. But alas, if relief ever comes it will undoubtedly be too little and too late. Ham bands will continue to be the domain of the solitary individual who has little to say of lasting value but an incredible amount of time in which to say it. Just another part of the electromagnetic wasteland shared by television and other forms of communication. If two meters is any indication, better we should all stick to microcomputers.... ■

Feature

GRAPHSUB: A PL/I-80 Graphics Package

by Robert S. Cahn

GRAPHSUB is a software package which was written to produce a high speed and high resolution graphics system. While a 256 by 256 system may be fine for animation, I am more interested in static images and the interplay between mathematics and the fine arts. This is an application which needs more resolution. Until recently, I had to use a graphics terminal but with the advent of intelligent graphics boards, it has become possible to do such work on a microcomputer. Using Digital Research's PL/I-80 and Scion's Microangelo MA-512 board I am able to move virtually all my work off the mainframe. This article will describe the software package and the graphics subsystem it uses. Since the software is modular you can modify this package to run on any CP/M system with a graphics sub-system by changing the lower level modules. Everybody has a personal definition of high resolution and I will not add my own. For my work, the Scion MA-512 Microangleo (MA) Board was adequate. It fit into a single slot on my S-100 system and made no demands on the host other than power and the use of two I/O ports. The resolution was 512 by 480 or about one-half the resolution of a Tek scope in either direction.

The board communicates with the host through a status port and a communication port. These are factory set to F1H and F0H but can be jumpered to other addresses. The communications to the Microangelo use a conventional handshaking arrangement. If the host desires to write to the MA it inputs the status port and checks the low bit to see if the transmit buffer is empty. When the buffer is clear the host outputs the data to the data port.

MA to host communications are similar. The MA comes with a variety of operating systems which are installed in ROM. The minimal system, Screenware Pak I, provides:

1. Clear Screen
2. Draw a Point
3. Draw a Vector
4. Draw an ASCII Character
5. Draw Crosshairs,

etc. An upgraded operating system called Screenware Pak II adds:

1. Relative Coordinates
2. Circle Generation
3. Area Flood
4. Macro Facilities.

I opted for the enhanced system with the extra commands. The commands for the MA system are downloaded from the host as a series of bytes. A command byte has the high bit set and is interpreted by the MA operating system which is stored in 6K of ROM. An example is the following short program written in Microsoft BASIC. The program will clear the screen and then draw a circle of radius 50 screen units with center at (240,256).

Listing 1

```
05  DEFINT X,I
10  X=136:GOSUB 10000:REM THIS IS THE CLEAR
    SCREEN CODE
20  X=141:GOSUB 10000:REM 141 MEANS DRAW A
    POINT
30  X=01:GOSUB 10000:REM THIS IS THE HIGH
    BYTE OF THE X COOR
40  X=00:GOSUB 10000:REM THIS IS THE LOW BYTE
    256=1*256+0
50  X=00:GOSUB 10000:REM 240 = 0*256 + 240
60  X=240:GOSUB 10000
70  X=201:GOSUB 10000:DRAW A CIRCLE
80  X=50:GOSUB 10000:REM SEND THE RADIUS
90  STOP
10000  I=INP(241)
10010  IF I<>2*INT(I/2) THEN 10000:REM LSB=1 IF
       BUSY
10020  OUT 241,X
10030  RETURN
20000  END
```

Though programming in BASIC is quick and easy to debug, the lack of local variables will quickly turn a longer program into a hopeless muddle. Further, the code executes too slowly. Even compiled BASIC is not fast enough for an application which generates thousands of plotting commands. Since I have plenty of memory on my system, my thoughts turned to PL/I.

As many benchmarks have shown, PL/I runs very fast. The catch is that the compilations are time consuming and the object modules are large.

The size problem is deceptive. PL/I-80 object modules are large because the full library is 41K. A single GET LIST will usually increase the size of the generated .COM file by 10K. There are dozens of library procedures needed for entering a simple YES or NO at run time. Yet, after the library procedures are added, one may add lots and lots of further I/O to a PL/I-80 program with only a modest increase in the size of the resulting .COM file. Tests showed that any program I use can be compiled in 35Kbytes or fewer.

The organization of GRAPHSUB

GRAPHSUB is a PL/I library which can be linked to any PL/I program. One must add the statement:

%INCLUDE B:GRAPHSUB.DCL;

to the main program if you are keeping your program files on disk B:. A listing of GRAPHSUB.DCL is shown below:

Listing 2

```
DCL   RESET ENTRY,
      CLEAR ENTRY,
      POINTA ENTRY (FIXED,FIXED),
      VECTA ENTRY (FIXED,FIXED),
      MOVEA ENTRY (FIXED,FIXED),
      CIRCLA ENTRY (FIXED,FIXED,FIXED),
      CHROUT ENTRY (CHAR(1)),
      LABEL ENTRY (FLOAT,FLOAT,
        CHAR(40 VARYING)),
```

```
INIT ENTRY (FLOAT,FLOAT,FLOAT,FLOAT),
WINDOW ENTRY
  (FLOAT,FLOAT,FLOAT,FLOAT),
CLIP ENTRY
  (FLOAT,FLOAT,FLOAT,FLOAT,FIXED),
SCALE ENTRY
  (FLOAT,FLOAT,FIXED,FIXED,FIXED),
VECTOR ENTRY (FLOAT,FLOAT),
POINT ENTRY (FLOAT,FLOAT),
MOVE ENTRY (FLOAT,FLOAT),
GRID ENTRY,
ARRSCL ENTRY (FIXED),
PLOT ENTRY (FIXED),
PLOT2 ENTRY (FIXED),

(X(300),Y(300)) STATIC EXTERNAL FLOAT;
```

Since all the procedures are external to the program using them, they must all be declared with the ENTRY attribute. If you are going to use only one or two routines you can declare them individually instead of including the file. The routines in GRAPHSUB are a hierarchically organized set of procedures. They are illustrated below:

Level 3 **Higher level graphics commands**
This level contains the routines GRID, ARRSCL, PLOT and PLOT2 allowing array graphing using only one procedure call.

Level 2 **Real world coordinate systems graphics.**
These routines create graphics using real world coordinates. Includes the procedures MOVE, POINT, VECTOR, CLIP, INIT, SCALE, WINDOW, LABEL.

Level 1 **Absolute or screen graphics**
This level will output graphics in screen coordinates. The routines include CHROUT, VECTA, POINTA, and MOVEA.

Level 0 **Communication with the Graphics Sub System**
These are the lowest level of routine which handles the actual communication with the MA board. These routines are called most often and are most effective when written in assembler. These routines are entirely device dependent. They include RESET, CLEAR, OUTBYT, OUTINT, and OUTCOO. Usually only RESET and CLEAR are called directly.

If it is desired to adapt GRAPHSUB to another system only level 0 and level 1 routines need be rewritten. The higher level procedures are usable on any system with vector primitives.

The procedures are linked into a PL/I library in the order shown. The linker in PL/I-80 can search a library using the S option looking for undefined entry points.

Example:

```
LINK B:PROGRAM,B:GRAPHSUB[S],
```

The linker will link the file B:PROGRAM.REL with the needed modules from the PL/I library and then search B:GRAPHSUB for any undefined entry points. It makes a single pass. If the procedure MOVE (world coordinates move) calls the procedure MOVEA (absolute move) it is imperative that the module MOVE appear in the library before MOVEA.

Use of GRAPHSUB

For a listing of GRAPHSUB you should refer to appendix A. There are 21 entry points and depending on your version of PL/I-80 it may take 20 separate compilations and assemblies to create all the needed .REL files. The .REL files will then need to be linked into a library with the high level routines first and the lower level routines last. It is only necessary to call the routine PLOT, however, to make conventional graphs.

Listing 3

```
PLTTST: PROC OPTIONS (MAIN);

%INCLUDE 'B:GRAPHSUB.DCL';

DCL (I,J) FLOAT,
      U FLOAT,
      V FLOAT,
      N FIXED;

DO J=1 TO 10;
/* ON EACH PASS THE PLOTTING WINDOW IS
SMALLER THAN BEFORE WITH A LARGER AND
LARGER BORDER AROUND THE PLOT */

CALL WINDOW(10*J,511-10*J,10*J,479-10*J);

N=0;

DO I = 0 TO 3.5 BY .1;
        X(N+1)=I;
        Y(N+1)=I*I;
        N=N+1;
        END;

CALL PLOT(N);

/* PLOT THE ARRAYS X(1),...,X(N)
  AND Y(1),...,Y(N) */

N=0;

DO I=1 TO 3.47 BY .25;
        X(N+1)=I;
        Y(N+1)=1/I;
        N=N+1;
        END;

CALL PLOT2(N);

/* PLOT2 PLOTS THE X AND Y ARRAYS IN THE AREA
  DETERMINED BY THE FIRST PLOT */

N=0;

DO I = 0 TO 3.5 BY .10;
        X(N+1)=I;
        Y(N+1)=EXP(I);
        N=N+1;
        END;

CALL PLOT2(N);

END;

END PLTTST;
```

This program plots the function y=x*x by storing the data in the arrays X and Y and then calling the procedure PLOT. PLOT first calls the routine ARRSCL to determine the size of the window needed to display the data. It then erases the screen using CLEAR and then draws coordinate axes using GRID. The axes are labeled by using OUTCHR to ►

output the coordinates converted to ASCII. The plotting cursor is MOVEd to the location X(1),Y(1) and then all subsequent points are connected using the procedure VECTOR. After plotting the first function the subsequent functions are plotted using PLOT2. This plot will be superimposed on the first one so as to display several separate functions which have the same range on the X axes.

At the other end of GRAPHSUB are the routines which handle communications with the MA board. These routines generally output either an 8-bit integer or a 16-bit integer to the board. Generally 8-bit integers will be thought of as unsigned numbers between 0 and 255. Characters will be treated as integers between 0 and 127. PL/I generally passes to a subprocedure not an argument but the address of the argument. This is call by reference.

Example.
The PL/I instruction CALL OUTBYT(X) will generate a CALL instruction. The DE register is a pointer to the address of the byte X to be output. OUTBYT then loads the address of X into the HL register and finally using indirect addressing loads the argument.

The OUTBYT and OUTINT procedures use a common argument loading routine. The names are declared public so that the linker will be able to recognize them. Names may be longer than six characters but only the first six characters can be used externally. If two programs were named OUTBYTE and OUTBYTES the PL/I library manager will refer to both as OUTBYT. Be careful, if you add new modules to the library, to observe this convention.

OUTBYT will output the lower 8 bits of an integer to the MA board without change. OUTINT performs a range check, however. Since it is used to transmit absolute coordinates it will change negative integers to 0 and integers over 511 to 511. The RESET procedure is the only other assembly language routine. It resets the MA by outputs to the command/status port.

In between level 3 and level 0 most of the work is done. The procedure WINDOW will define the part of the plotting screen which is to be used for graphics.

Example
To confine the graphics to the top of the page we would

CALL WINDOW(0,511,240,479);.

To leave a boundry of 50 screen units around the boundry we would

CALL WINDOW(50,461,50,429);.

Once the graphics window has been defined we must designate what world coordinates are to be used in the window. This is done by calling INIT. INIT(-6,6,5,5) will allow plotting data with $-6 <= x <= 6$ and $-5 <= y <= 5$ inside the plotting window. The other routines allow the plotting of points, lines, vectors, circles, and text.

Variable usage

Once established, the screen limits are passed to the various procedures using EXTERNAL STATIC variables. This storage class is much like the FORTRAN COMMON variables. Such variables are global among all procedures where they are declared. The WINDOW procedure uses the variables XMIN and XMAX for the X coordinate and YMIN to YMAX for the Y coordinate. The INIT routine uses the fixed variables SXMIN, SXMAX, SYMIN and SYMAX. The transfer of arrays to the higher level graphing procedures is a problem. PL/I-80 does not allow arrays to be passed to procedures using the declaration,

DCL X(*) FLOAT;

one of the more pleasant features of full PL/I. One solution is to use EXTERNAL STATIC arrays. GRAPHSUB. DCL declares the real arrays X and Y and uses them to pass data plotting procedures. This is adequate for most purposes.

Another approach is to play tricks with pointers. A pointer is used in PL/I based storage to refer indirectly to data. Pointers can be qualified by a statement such as,

P -> X(1);

but cannot be incremented or decremented. If the array is stored linearly, however, then the address of each subsequent element of the array is a fixed offset from the address of its predecessor. If X is an array of type float,

X(1),X(2),X(3),. . .,X(N),

then the address of X(2) is four greater than the address of X(1), the address of X(3) is eight greater than address of X(1), etc. It is then possible to pass to a procedure a pointer to the begining of the array and the size of the array and have the data retrieved.

Results

We illustrate the use of GRAPHSUB by an example taken from the fine arts. A conformal mapping is a transformation of the plane which preserves angles but changes distances. These functions have been a mainstay of electrical engineering. We will use functions that are expressed in complex notation. Instead of thinking of a point as having coordinates (x,y) we think of this point as

$$z = x + iy$$

with i being the square root of -1. It is possible to add, subtract, multiply, and divide complex numbers.

Listing 4

```
CONFRM: PROC OPTIONS(MAIN);

%INCLUDE 'B:GRAPHSUB.DCL';

DCL   XX(1000) FLOAT; /*THESE ARE THE
        COORDINATES BEFORE MAPPING */
DCL   YY(1000) FLOAT;
DCL   FLAG(1000) FIXED BINARY (7); /* A POINT OR
        A DRAW */
DCL   (U,V) FLOAT; /*THESE ARE THE RETURNED
        COMPLEX COORDINATES */
DCL   (X1,Y1) FLOAT;/* WORKING VARIABLES */
DCL   (XOFF,YOFF,SIZE) FLOAT; /* THESE GOVERN
        IMAGE SIZE AND LOC */
DCL   (I,J) FIXED; /* WORKING VARIABLES */
DCL   AGAIN FIXED STATIC INITIAL(1);
DCL   NAME CHAR(13) VARYING;/* NAME OF THE
        FILE */

DCL   (DATA) FILE;

ON UNDEFINEDFILE(DATA) BEGIN;
```

```
        PUT SKIP LIST('THAT FILE IS NOT ON THE
          DISK');
        GO TO QUERY;
        END;
QUERY: PUT SKIP LIST ('WHAT IS THE NAME OF
  THE DATA FILE?');
GET LIST (NAME);
OPEN FILE (DATA) STREAM INPUT TITLE(NAME) ;

/ * * * * * MAIN PROCEDURE * * * * * /

I=1;
ON ENDFILE (DATA) GOTO NEXT;
DO WHILE (AGAIN ↑= 0);
        GET FILE(DATA)
EDIT(FLAG(I),XX(I),YY(I))(F(1),E(6),E(6));
        I=I+1;
        END;
NEXT:CLOSE FILE(DATA);
DO WHILE (AGAIN ↑=0);
ON ERROR(1) BEGIN;
        PUT SKIP LIST('DATA NOT CORRECT TYPE');
        GO TO PARAM;
        END;
PARAM: PUT LIST ('WHAT SIZE FOR THIS IMAGE?');
GET LIST (SIZE);
PUT LIST ('WHAT X AND Y OFFSETS?');
GET LIST (XOFF,YOFF);
CALL WINDOW(0,480,0,480);
CALL INIT(-8,8,-8,8);
CALL CLEAR;
DO J=1 TO I-1;
        X1=XX(J)*SIZE+XOFF;
        Y1=YY(J)*SIZE+YOFF;
        CALL TRANS(X1,Y1,U,V);
        IF FLAG(J)=0 THEN CALL POINT(U,V);
        ELSE CALL VECTOR(U,V);
        END;
PUT LIST('DO YOU WISH TO DRAW ANOTHER IM-
AGE? 1=YES 0=NO');
GET LIST(AGAIN);
END;
TRANS: PROCEDURE (R,S,T,W);
DCL (R,S,T,W,DEN) FLOAT;
DEN = R*R+S*S;
IF DEN<.0001 THEN DO;
        T=100;
        W=100;
        END;
ELSE DO;
        T=R-(S/DEN);
        W=S+(R/DEN);
        END;
RETURN;
END TRANS;
END CONFRM;
```

In Listing 4 the function used is $w = z + i/z$. The function is used to transform a digitized image which is stored on disk as a sequential file. In Figure 2 you have the result of a series of these transformations. The "distortions" represent a new method for looking at space and using perspective. The plots are produced as quickly as on a much larger timesharing system.

PL/I-80 and Scions board have done the job for me.

APPENDIX

Program Listings

The following listings are referred to in the text.

Depending on your version of PL/I-80 you will have to do separate compilations and link the result together by the library manager or you can enclose all of the PL/I programs in a single dummy program and then link the assembly language routines.

Listing A.1

```
PLOT:PROC (N);

DCL N FIXED;
DCL (X(300),Y(300)) EXTERNAL STATIC FLOAT;
DCL I FIXED;
DCL CLEAR ENTRY;
DCL ARRSCL ENTRY (FIXED);
DCL GRID ENTRY;
DCL POINT ENTRY (FLOAT,FLOAT);
DCL VECTOR ENTRY (FLOAT,FLOAT);

IF (N<2 ! N>300) THEN DO;
        PUT LIST('ERROR IN PLOT ROUTINE');
        RETURN;
        END;

CALL CLEAR;
CALL ARRSCL(N);
CALL GRID;
CALL POINT (X(1),Y(1));

DO I=2 TO N;
        CALL VECTOR (X(I),Y(I));
        END;

RETURN;

END PLOT;
```

Listing A.2

```
PLOT2:PROC (N);

DCL N FIXED;
DCL (X(300),Y(300)) EXTERNAL STATIC FLOAT;
DCL I FIXED;

DCL POINT ENTRY (FLOAT,FLOAT);
DCL VECTOR ENTRY (FLOAT,FLOAT);

IF (N<2! N>300) THEN DO;
        PUT LIST('ERROR IN PLOT2 ROUTINE');
        RETURN;
        END;

CALL POINT (X(1),Y(1));

DO I=2 TO N;
        CALL VECTOR (X(I),Y(I));
        END;

RETURN;

END PLOT2;
```

►

Listing A.3

```
ARRSCL:PROC (N);

DCL N FIXED;
DCL (X(300),Y(300)) EXTERNAL STATIC FLOAT;
DCL (XMAX,XMIN,YMAX,YMIN) EXTERNAL STATIC FLOAT;
DCL (B,T) FLOAT;
DCL (I,J) FIXED;

/* THIS ROUTINE WILL SIZE THE ARRAY */

B = X(1);T = X(1);

DO I = 1 TO N;
      IF (X(I)>T) THEN T = X(I);
      IF (X(I)<B) THEN B = X(I);
      END;

XMAX = CEIL(T);
XMIN = FLOOR(B);

B = Y(1);T = Y(1);

DO I = 1 TO N;
      IF (Y(I)>T) THEN T = Y(I);
      IF (Y(I)<B) THEN B = Y(I);
      END;

YMAX = CEIL(T);
YMIN = FLOOR(B);

RETURN;
END ARRSCL;
```

Listing A.4

```
GRID: PROCEDURE;

DCL VECTOR ENTRY (FLOAT,FLOAT);
DCL MOVE ENTRY (FLOAT,FLOAT);
DCL CHROUT ENTRY (CHAR(1));

DCL (XMIN,XMAX,YMIN,YMAX) FLOAT EXTERNAL STATIC;

DCL (X,Y) FLOAT;
DCL C CHAR(9) VARYING;
DCL LETTER CHAR(1);
DCL (I,J) FIXED;

DO X = XMIN TO XMAX;
      CALL MOVE(X,YMIN);
      CALL VECTOR (X,YMAX);
      END;

DO Y = YMIN TO YMAX;
      CALL MOVE(XMIN,Y);
      CALL VECTOR(XMAX,Y);
      END;

DO X = XMIN TO XMAX;
      CALL MOVE(X,YMIN);
      I = X;
      C = CHARACTER(I);
      LETTER = '0';J = 1;
      DO WHILE ( J<=LENGTH(C) & LETTER ↑= ' . ');
            LETTER = SUBSTR(C,J,1);
            IF (LETTER ↑= ' ') THEN CALL
            CHROUT(LETTER);
            J = J + 1;
            END;
      END;

DO Y = YMIN TO YMAX;
      CALL MOVE(XMIN,Y);
      I = Y;
      C = CHARACTER(I);
      LETTER = '0';J = 1;
      DO WHILE (J <= LENGTH(C) & LETTER ↑= ' . ');
            LETTER = SUBSTR(C,J,1);
            IF (LETTER ↑= ' ') THEN CALL CHROUT(LETTER);
            J = J + 1;
            END;
      END;

RETURN;

END GRID;
```

Listing A.5

```
VECTOR:PROC (X,Y);

/* TEST OF PROBLEM GET RID OF CLIP */
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);

/* THIS IS WHERE THE WORK GETS DONE */

      CALL SCALE(X,Y,IX,IY,FLAG);
      CALL VECTA(IX,IY);

XPREV = X; YPREV = Y;

RETURN;

END VECTOR;
```

Listing A.6

```
POINT: PROC (X,Y);

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

DCL MOVEA ENTRY(FIXED,FIXED);

CALL SCALE(X,Y,IX,IY,FLAG);

IF (FLAG = 0) THEN DO;
      CALL POINTA(IX,IY);
      PVFLAG = 0;
      END;
ELSE PVFLAG = 1;

XPREV = X; YPREV = Y;
RETURN;

END POINT;
```

Listing A.7

```
MOVE: PROC (X,Y);

DCL (X,Y) FLOAT;
DCL (IX,IY,FLAG) FIXED;
DCL (XPREV,YPREV) FLOAT STATIC EXTERNAL;
DCL PVFLAG FIXED STATIC EXTERNAL:
DCL SCALE ENTRY (FLOAT,FLOAT,FIXED,FIXED,FIXED);
DCL MOVEA ENTRY (FIXED,FIXED);

CALL SCALE(X,Y,IX,IY,FLAG);

IF (FLAG = 0) THEN PVFLAG = 0
ELSE PVFLAG = 1;

CALL MOVEA(IX,IY);

XPREV = X; YPREV = Y;
RETURN;

END MOVE;
```

Listing A.8

```
CLIP: PROC (X,Y,XC,YC,PLOT);

DCL (X,Y,XC,YC) FLOAT;
DCL (XPREV,YPREV) STATIC FLOAT EXTERNAL;

DCL PVFLAG STATIC FIXED EXTERNAL;
/*PVFLAG = 0 PREV POINT INSIDE WINDOW*/

DCL PLOT FIXED;
/*IF PLOT = 0 THEN WE PLOT THE POINT ELSE WE OMIT */

DCL (XMIN,XMAX,YMIN,YMAX) STATIC FLOAT EXTERNAL;
DCL (BX,BY) FLOAT;
DCL INFLAG FIXED;

DCL MOVE ENTRY(FLOAT,FLOAT);

IF (X>=XMIN & X<=XMAX & Y>=YMIN & Y<=YMAX)
   THEN INFLAG = 0; ELSE INFLAG = 1;

PLOT = 0

IF (INFLAG = 0 & PFVLAG = 0) THEN DO;
        XC = X;
        YC = Y;
        RETURN;
        END;

If (INFLAG = 1 & PVFLAG = 1) THEN DO:
        XC = X;
        YC = Y;
        PLOT = 1;
        RETURN;
        END;

IF (INFLAG = 1 & PVFLAG = 0) THEN DO;
        XC = BX;
        YC = BY;
        PVFLAG = 1; /* SAVE THE INFLAG IN PVFLAG */
        RETURN;
        END;

IF (INFLAG = 0 & PVFLAG = 1) THEN DO;
        CALL BOUNDRY(BX,BY);
        CALL MOVE(BX,BY);
        XC = X;
        YC = Y;
        PVFLAG = 0;
        RETURN;
        END;

BOUNDRY:PROC (XB,YB);

DCL (INX,INY,OUTX,OUTY)FLOAT;
DCL (XB,YB) FLOAT;

IF INFLAG = 0 THEN DO;
        INX = X;
        INY = Y;
        OUTX = XPREV;
        OUTY = YPREV;
        END;
ELSE DO;
        INX = XPREV;
        INY = YPREV;
        OUTX = X;
        OUTY = Y;
        END;

IF OUTX<XMIN THEN DO;
        YB = OUTY + (XMIN – OUTX)*(INY – OUTY)/(INX – X);
        IF YB<=YMAX & YB>=YMIN THEN DO;
            XB = XMIN;
            RETURN;
            END;
        END;

IF OUTY<YMIN THEN DO;
        XB = OUTX + (INX – OUTX)*(YMIN – OUTY)/(INY – OUTY);
        IF XB>=XMIN & XB<=XMAX THEN DO:
            YB = YMIN;
            RETURN;
            END;
        END;

IF OUTX>XMAX THEN DO;
        YB = INY + (OUTY – INY)*(XMAX – INX)/(OUTX – INX);
        IF YB>=YMIN & YB<=YMAX THEN DO;
            XM = XMAX;
            RETURN;
            END;
        END;

IF OUTY>YMAX THEN DO;
        XB = INX + (OUTX – INX)*(YMAX – INY)/(OUTY – INY);
        IF XB>=XMIN & XB<=XMAX THEN DO;
            YB = YMAX;
            RETURN;
            END;
        END;

END BOUNDRY;

END CLIP;
```

Listing A.9

```
SCALE:PROC(X,Y,IX,IY,FLAG);

DCL (XMIN,XMAX,YMIN,YMAX) FLOAT STATIC EXTERNAL;
DCL (SXMIN,SXMAX,SYMIN,SYMAX) FLOAT STATIC EXTERNAL;

DCL (X,Y) FLOAT;
DCL (IX,IY,FLAG)FIXED;

FLAG = 0;

IF (X>XMAX) THEN DO;
        IX = FIXED(SXMAX);
        FLAG = 1;
        END;
ELSE IF (X<XMIN) THEN DO;
            IX = FIXED(SXMIN);
            FLAG = 1;
            END;
  ELSE IX = FIXED(SXMIN + (X-XMIN)*(SXMAX-SXMIN)/
  (XMAX-XMIN));

IF (Y>YMAX) THEN DO;
        IY = FIXED(SYMAX);
        FLAG = 1;
        END;
ELSE IF (Y<YMIN) THEN DO;
        IY = FIXED(SYMIN);
        FLAG = 1;
        END;
  ELSE IY = FIXED(SYMIN + (Y-YMIN)*(SYMAX-SYMIN)/
  (YMAX-YMIN));

RETURN;

END SCALE;
```

Listing A.10

```
WINDOW:PROC(X1,X2,Y1,Y2);

DCL (SXMIN,SXMAX,SYMIN,SYMAX) FLOAT STATIC EXTERNAL;

DCL (X1,X2,Y1,Y2) FLOAT;

/* X1 AND Y1 ARE THE MINIMA AND X2 AND Y2 THE MAXIMA */

IF (X2<=X1 | X1>500 | X2 <10 ) THEN DO;
        SXMIN = 0;
        SXMAX = 511;
        END;
```

```
ELSE DO;
        SXMIN = MAX(0,X1);
        SXMAX = MIN(511,X2);
        END;

/* WE LIMIT THE SCREEN COORDINATES TO THE PHYSICAL
DIMENSIONS */

IF (Y2<=Y1 | Y1>500 | Y2<10) THEN DO;
        SYMIN = 0;
        SYMAX = 480;
        END;
ELSE DO;
        SYMIN = MAX(0,Y1);
        SYMAX = MIN(Y2,480);
        END;

RETURN;

END WINDOW;
```

Listing A.11

```
LABEL: PROCEDURE(X,Y,MESAGE);

DCL MOVE ENTRY (FLOAT,FLOAT);
DCL CHROUT ENTRY (CHAR(1));

DCL (X,Y) FLOAT;
DCL MESAGE CHAR(40) VARYING;
DCL LETTER CHAR(1);
DCL (I,J) FIXED;

CALL MOVE(X,Y);

DO I = 1 TO LENGTH(MESAGE);
        LETTER = SUBSTR(MESAGE,I,1);
        CALL CHROUT(LETTER);
        END;

RETURN;

END LABEL;
```

Listing A.12

```
CHROUT: PROCEDURE (LETTER);

DCL OUTBYT ENTRY(FIXED(15));
DCL CODE FIXED;
DCL LETTER CHAR(1);

CODE = 152;
CALL OUTBYT(CODE);

CODE = RANK(LETTER);
CALL OUTBYT(CODE);

RETURN;

END CHROUT;
```

Listing A.13

```
INIT: PROC (XB,XT,YB,YT);

DCL (XB,XT,YB,YT) FLOAT;
DCL (XMIN,XMAX,YMIN,YMAX) FLOAT STATIC EXTERNAL;

XMIN = XB;
XMAX = XT;
YMIN = YB;
YMAX = YT;

RETURN;

END INIT;
```

Listing A.14

```
VECTA: PROCEDURE (IX,IY);

DCL OUTBYT ENTRY(FIXED(15));
DCL OUTCOOR ENTRY(FIXED(15),FIXED(15));
DCL (CODE,IX,IY) FIXED;

CODE = 145;

CALL OUTBYT(CODE);
CALL OUTCOOR(IX,IY);

RETURN;

END VECTA;
```

Listing A.15

```
POINTA: PROCEDURE (IX,IY);

DCL OUTBYT ENTRY(FIXED(15));
DCL OUTCOOR ENTRY(FIXED(15),FIXED(15));
DCL (CODE,IX,IY) FIXED;

CODE = 141;

CALL OUTBYT(CODE);
CALL OUTCOOR(IX,IY);

RETURN;

END POINTA;
```

Listing A.16

```
MOVEA: PROCEDURE (IX,IY);

DCL OUTBYT ENTRY(FIXED(15));
DCL OUTCOOR ENTRY(FIXED(15),FIXED(15));
DCL (CODE,IX,IY) FIXED;

CODE = 142;

CALL OUTBYT(CODE);
CALL OUTCOOR(IX,IY);
CALL OUTBYT(CODE);
CALL OUTCOOR(IX,IY);

RETURN;

END MOVEA;
```

Listing A.17

```
OUTCOOR: PROCEDURE (IX,IY);

DCL OUTINT ENTRY(FIXED(15));

DCL (IX,IY) FIXED;
CALL OUTINT(IX);
CALL OUTINT(IY);

RETURN;

END OUTCOOR;
```

Listing A.18

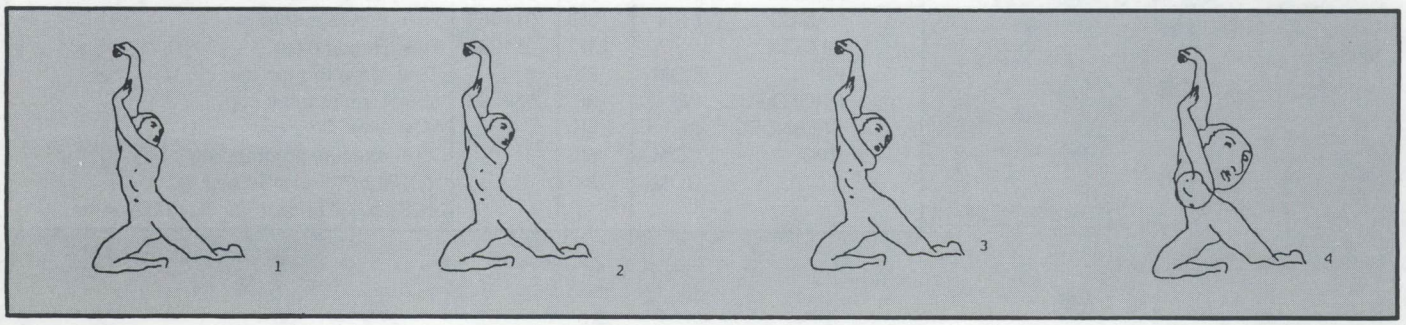
```
CLEAR:      PROCEDURE;

DCL OUTBYT ENTRY(FIXED(15));
DCL X FIXED;

X = 136;
CALL OUTBYT(X);
RETURN;

END CLEAR;
```

Listing A.19

```
TITLE 'PL/I TO MICROANGELO INTERFACE SUBROUTINES'
;
;           THE FIRST OF THESE ROUTINES WILL OUTPUT A
;           BYTE
;           THE PARAMETER SHOULD BE A NUMBER RATHER
;           THAN A BIT STRING
;           ONLY THE FIRST BYTE IS PASSED TO THE
;           MICROANGELO
;
;           THE SECOND ROUTINE OUTPUTS A BINARY
;           INTEGER WHICH IS NORMALIZED
;           TO LIE BETWEEN 0 AND 511 WITH ENDS INCLUDED.
;
;           NEITHER ROUTINE RETURNS ANY DATA TO THE
;           CALLING PROGRAM
;
;
;           PARAMETER LOADER SUBS
;
GETP1:
            MOV     E,M             ;LOW HALF
            INX     H
            MOV     D,M
            INX     H
            XCHG
            MOV     C,M
            XCHG
            RET
;
;
;           GET A WHOLE 16 BIT NUMBER
;
GETP2:
            CALL    GETP1
            INX     D
            LDAX    D
            MOV     B,A
            RET
;
;           SENDS THE BYTE IN THE A REGISTER TO THE
;           MICROANGELO WHEN IT GETS THE OK.
;           LEAVES THE OTHER REGISTERS UNDISTURBED.
;
SEND:       PUSH    PSW             ;SAVE THE A REGISTER
            IN      241             ;IS IT READY TO
                                    ;RECEIVE A BYTE
            ANI     1               ;CHECK THE LOW BIT
            JNZ     SEND            ;BACK WE GO IN A
                                    ;TIGHT LOOP
            POP     PSW
            OUT     240
            RET
;
;***************************************************
;
;           FUNCTION OUTBYT(X) X FIXED BINARY 0<X<255 OR
;           X BIT(8)
;
;           PUBLIC OUTBYT
;
OUTBYT      CALL    GETP1
            MOV     A,C
            CALL    SEND
            RET
;
;***************************************************
;
;           FUNCTION OUTINT(X) X FIXED BINARY. IF X<0 THEN
;           0 IS OUTPUT
;           AND IF X>511 THEN 511 WILL BE OUTPUT.
;
;           PUBLIC OUTINT
;
OUTINT      CALL    GETP2
            MOV     A,B
TEST1       ANI     80H             ;TEST1 ASKS IF THE
                                    ;NUMBER IS NEGATIVE
                                    ;IN
            JZ      TEST2           ;2S COMPLEMENT
            LXI     B,0000H         ;IF IT WAS NEGATIVE
                                    ;LETS MAKE IT 0
            JMP     OUTIN2
TEST2       MOV     A,B             ;PUT THE B REG BACK
                                    ;IN THE A
            ANI     7EH             ;IS B>1? IF SO YOU
                                    ;MUST CLIP
            JZ      OUTIN2          ;IF IT PASSES TEST 2
            LXI     B,01FFH         ;PUT A 511 IN THE BC
                                    ;REGISTER
OUTIN2      MOV     A,B             ;LOAD THE HIGH BYTE
                                    ;IN A
            CALL    SEND
            MOV     A,C             ;LOAD IN THE LOW
                                    ;BYTE
            CALL    SEND
            RET                     ;BACK TO PL/I
;
END
```

Listing A.20

```
TITLE 'RESET MICROANGELO SUBROUTINE'
;
;           THIS SUBROUTINE HAS NO PARAMETERS BUT
;           RESETS THE MA BOARD
;
;
;           PUBLIC RESET
;
RESET       MVI     A,1
            OUT     241
            MVI     A,0
            OUT     241
            RET
            END
```



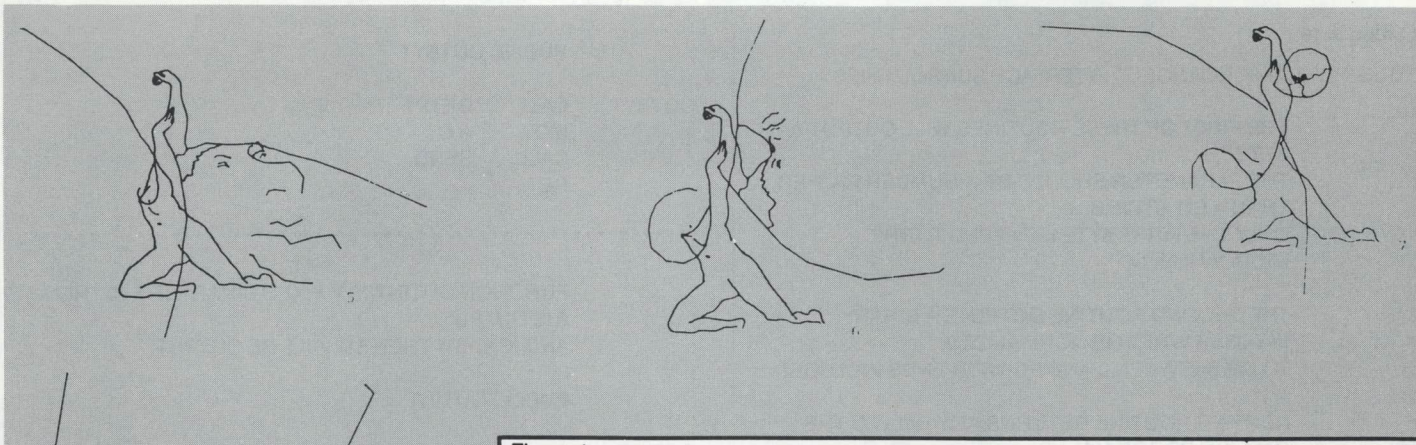


Figure 1. (continued from page 16)

* Simplifile * Disk Left: 65K File: 7 of 16			Sort Exit	Mult FLst	Current: B Backup: A >Date: 042883 User: 0 Id: TEXT1
CMD	**Name**	**Type**	**Size**	**ChgDte**	**Description**
	—TEXT1	DID	2K	042883	Simplifile parameters, directory, ID
	ARTICLE	LLS	18K	050183	Simplifile article for *Lifelines*
	ARTICLE2	AT	9K	062183	Article on The Champion for Ashton-Tate
	CURVIT2	CES	4K	040183	Updated resume with art config. for Diablo
	CURVIT2	DFT	3K	040183	Updated resume config. for Epson
	CURVIT2	FF	3K	040183	Experiment—config. for Fancy Font & Epson
=>	ELGTPRT		20K		Preliminary text for book project
	LOGO3	ART	2K		Wordprocessor art "NOLO PRESS NEWS"
	HARRIS	WP	7K	032583	Portions of fictional work—on Panlantir WP
	NAMES	PRN	5K	011583	Address list
	OLDISK		21K	051283	Fict—"Ancient Computer Disk Deciphered!"
	PIP	COM	8K		
	PROLOGUE	WP	13K	052783	Fiction—on Palantir WP
	SALSNDR2	ART	4K		Experiment—Word Processor Artwork
	STAT	COM	6K		
	X	COM	3K		Directory utility from Workman & Assoc.

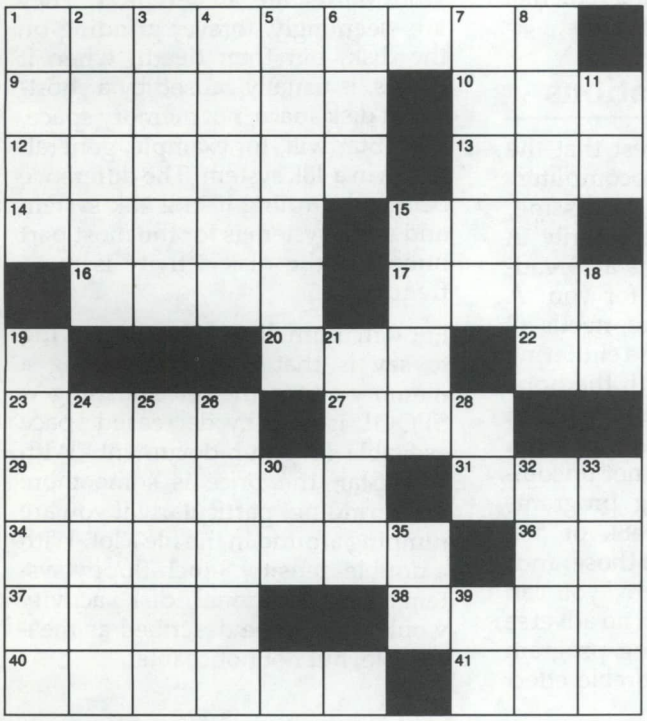
Figure 2.

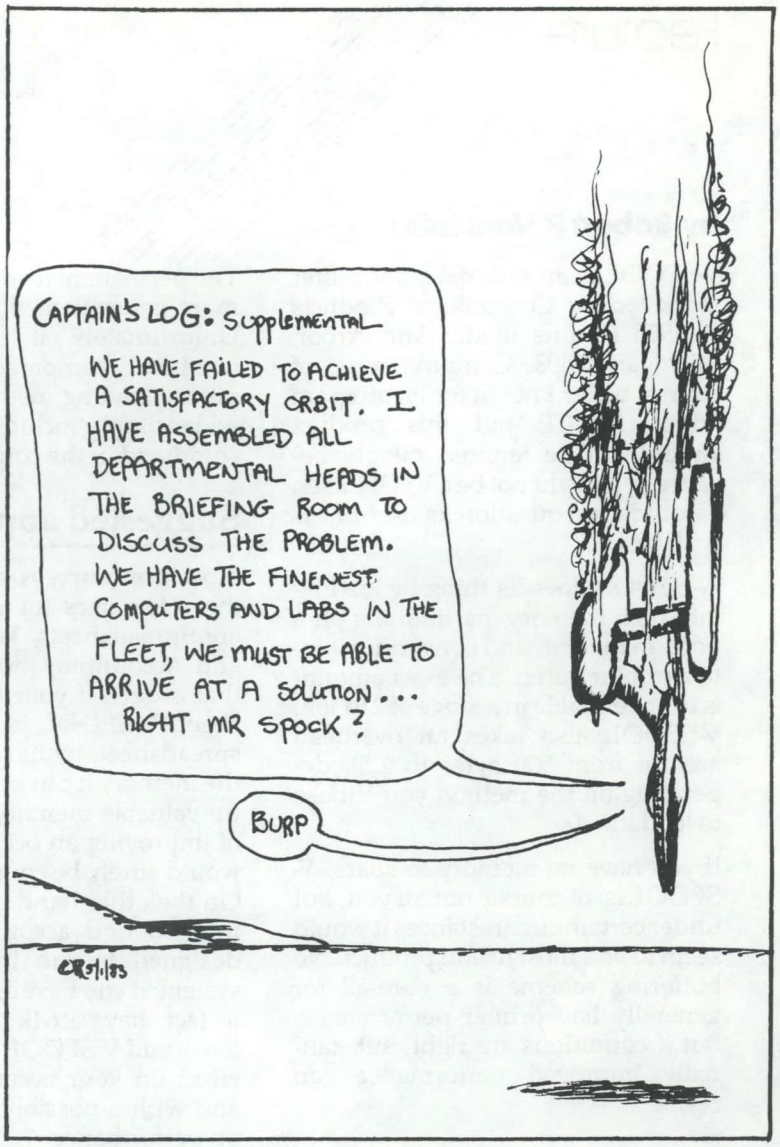
* Simplifile * Disk Left: 65K File: 7 of 16			Sort Exit	Mult FLst	Current: B Backup: A >Date: 042883 User: 0 Id: TEXT1
CMD	**Name**	**Type**	**Size**	**ChgDte**	**Description**
	—TEXT1	DID	2K	042883	
	ARTICLE	LLS	18K	050183	? = Display File CMD's
	ARTICLE2	AT	9K	062183	B = Backup file
	CURVIT2	CES	4K	040183	C = Copy file
	CURVIT2	DFT	3K	040183	E = Erase file
	CURVIT2	FF	3K	040183	G = Goto file
	ELGTPRT		20K		L = List file
	LOGO3	ART	2K		M = Mark file for Mult
	HARRIS	WP	7K	032583	N = Next screen
	NAMES	PRN	5K	011583	P = Previous screen
	OLDISK		21K	051283	R = Rename file
	PIP	COM	8K		T = Make this top row
	PROLOGUE	WP	13K	052783	U = User number copy
	SALSNDR2	ART	4K		V = View file
	STAT	COM	6K		X = eXecute program
	X	COM	3K		<CTRL>P = List Screen
					<ESC> = "escape" or Go to Disk Area

Crossword Puzzle

by Crescent Varrone

This month we begin our crossword puzzle, which includes computer jargon, computer companies, and various obscure words. The solution will appear in October Lifelines.





Across

1. With 19 Down, large retail dealer of *Lifelines*.
9. Lasso.
10. Lumberjack's tool.
12. Tristram's love.
13. Separate area or div.
14. Yearned.
15. Present for Dad.
16. Bolger's stuffing (obs. sp.).
17. __ltronix, Canadian retail dealer of *Lifelines*.
20. Opposite of ACK.
22. Trdmk., for short.
23. Evolutionary operations.
27. D-lysurgic acid.
29. Crab __.
31. Display attention bits.
34. Lattice.
36. __ mein.
37. Trick.
38. Computers store this.
40. Glazier's diamond.
41. __ scan.

Down

1. High-level list-processing language.
2. Fertile place in the desert.
3. ". . .on the Western __."
4. Freemason's doorkeeper.
5. __ Zee, Dutch sea-arm.
6. 8 homophone.
7. Number-system base.
8. Anticipate.
11. Movie idol, summer '82.
15. Siberian goats.
18. Dorothy's Auntie __.
19. See 1 Across.
21. Jai __.
24. *Deum __ de Deo vero.*
25. Corpulent.
26. Weakling.
28. Ministerial letters.
30. This mag.
32. __ Romeo.
33. To start up a system.
35. Affirmative, to Zurbarán.
39. Method of machine control by digital instruct.

V-SPOOL Reviewed

by Robert P. VanNatta

V-SPOOL is an internal print buffer marketed by CompuView Products of 1955 Pauline Blvd., Ann Arbor, Michigan 48103. CompuView is, of course, better known for its program editor, VEDIT; but this product should not be ignored merely because you might not be a VEDIT user. I would even question its usefulness with VEDIT.

V-SPOOL does its thing by loading into high memory, partitioning off a portion thereof, and converting it into a printer buffer. The exact amount is user definable in a range of 2 to 16k. V-SPOOL also takes an overhead area of from 500 bytes to 2.5k depending on the method you choose to load it.

If you have no memory to spare, V-SPOOL is, of course, not for you; but under certain circumstances it would seem to be a most useful product. No buffering scheme is a cure-all for generally bad printer performance, but if conditions are right, substantially improved performance can result.

Use is easy

One of the most pleasant things about V-SPOOL is its ease of use. It is delivered in three files. One is the utility itself, called SPOOL.COM. The other files are a configuration routine (SPOOLIN.COM) and a spool remover (UNSPOOL.COM). SPOOLIN can be used to adjust the default size of the buffer to 2, 4, 8, or 16k. Once you have decided on the size you want, and have run the configuration program, the buffer can be loaded in two ways: (1) You can simply enter "SPOOL," in which case the buffer will load and remain there until you either execute a cold boot or run 'UNSPOOL', or (2) you can load the spooler on the command line with the program you are going to run. Thus "SPOOL WORDSTAR" will load the spooler and execute your program called 'WORDSTAR.COM'. The trade-off in method is one of convenience for memory. The permanent loading procedure is more convenient (load and forget). Unfortunately it takes 2k extra overhead compared to the temporary loading method accomplished by simply including it as the first command in the command line.

Suggested applications

Most user surveys suggest that the three top uses for microcomputers are Spreadsheets, Word Processing, and Accounting (not necessarily in that order). If your bag is a spreadsheet, V-SPOOL is not for you. A spreadsheet, by its nature, needs all the memory it can get, and cluttering up valuable memory with the hope of improving an occasional printout would surely be counter-productive. On the other hand, it is not uncommon to find accounting programs designed to run in a 48k or 56k system; if you have one of those, and, in fact, have a 64k system, you can likely add V-SPOOL with no adverse effect on your accounting program and with a possibly favorable effect on performance.

When it comes to word processing programs, the cost-benefit is a little harder to calculate. An accounting program will either run (in which case the spooler is probably useful) or crash with an out-of-memory error. By contrast, most word processing programs contain subtle routines to adjust the size of the text buffer to match the amount of memory available. It goes without saying that larger text buffers are more pleasant to work with than smaller ones, but the trade-off is one of degree.

The performance of any word processing program will be degraded somewhat by reduced memory availability; however, some are affected more than others. Editors tend to fall in the class of either being memory oriented or file oriented. Spellbinder (by Lexisoft) and Vedit (by CompuView) are typical of the memory editor type. They tend to be small, very powerful, and very fast. They also tend to fall apart (at least in performance and convenience, if not literally) when the file gets larger than the available memory buffer. Of the file-oriented editors, WordStar and WordMaster are archetypical. They are, seemingly, forever grinding on the disk, but their death, when it comes, is usually caused by a shortage of disk space, not memory space. WordStar will, for example, generally run in a 48k system. The difference between running it in a 48k system and a 64k system is for the most part only that the disk activity is more frequent.

Enough mumbling! What I am trying to say is that if you are using a memory editor, the price of using V-SPOOL is paid by decreased space available for your document. With WordStar, the price is some more disk grinding, particularly if you are jumping around in the file a lot. With a double-density, 8-inch floppy system, the additional disk activity would have to be described as measurable, but not noticeable.

Why a Spooler, anyway?

As described above I would characterize V-SPOOL not so much a spooler as it is a printer buffer. Its primary usefulness is under circumstances where you desire to edit and print at the same time. This implies background printing. However, presently I shall describe one hardware combination that is benefited even if your only desire is to print. This writer's experience with background printing over the years has been rather dismal. This is essentially because most of the printers that I have used have required so much time to get their data, that an effort to edit and print at the same time usually resulted in getting neither done. As an aside, it should be mentioned that a notable exception to this rule has been the venerable (and now somewhat dated) Texas Instruments Model 810 (TI-810). The TI-810, as well as some other of the more ex-

pensive dot matrix printers, has supported the 9600 baud serial interface which is seemingly fast enough to make a background print facility work even without a buffer.[1]

Anyhow, the point I am trying to make is that with V-SPOOL the background print facility of WordStar becomes viable, even with a slow printer.

Problems

It seems as if all good things have a catch somewhere. A spooler, by its nature, must intercept the printer output. V-SPOOL assumes that the printer output is being sent though the CP/M list device. This implies that if you are by-passing the CP/M list device with a custom printer driver or the WordStar port driver, you are out of luck. Indeed, Lifeboat CP/M Version 2.25d, coupled with WordStar version 3.0, ignore V-SPOOL even when purportedly installed to use the list device. On other programs, Lifeboat CP/M seems to function normally.

With respect to ATON CP/M version 2.23 (Aton International, 260 Brooklyn Ave., San Jose, CA 95128) for the TRS 80 Model II/12/16, I found that it worked properly (even with WordStar) in the standard 64k mode but would not work at all with the bank-switched 128k version.

V-SPOOL checks out all right under Pickles and Trout CP/M 2.2e for the TRS-80. I can also report for the benefit of those of you who are running TRISOFT CP/M-68k (4102 Avenue G, Austin, TX 78751), on the Motorola 68000 side of their Model 16; V-SPOOL works with CP/M-68k as well. (Someday maybe there will even be a program or two to run with CP/M-68k as well.)[2] The reason that V-SPOOL works on the TANDY 16 is not because it is magically a 16-bit program, but rather because CP/M-80 controls the I/O for the MC68000 on the Model 16, and V-SPOOL is logically part of the I/O. Only time will tell whether history will categorize the Model 16 as a cat or a dog, but there is something to be said for a design that plugs an MC68000 into a Z-80 in such a way that an existing 8-bit operating system can effectively be made a slave doing faceless tasks of managing the input and output.

One tasteless software protection scheme bites the dust

V-SPOOL is delivered on a standard 8-inch floppy with instructions that direct you to put the master disk into drive B: and attempt to load the spooler, before you go through the routine of making a backup. I hardly ever read instructions before I start playing with a new program that I get in, so I was never burdened with the decision of whether to follow those instructions or not. It was only a few days later that I began to appreciate the brilliance of my habit of making a backup first and playing with the product later. It seems that the clever fellows down under (V-SPOOL was written in Australia) believe that it is immoral to have more than one operating system for your computer. Accordingly, they have jerry-rigged a mousetrap on V-SPOOL that latches onto the serial number of the first version of CP/M that is configured with it. If you have followed their setup instructions, this modification will actually be written back to the master disk, preventing you from using this product with any other operating system. To make a long story short, it wasn't very long until V-SPOOL was accusing me of being an outright software pirate. It may be that morals or laws are different in Australia, but I have never really understood that it was either immoral or illegal to have several legitimate, unpirated operating systems for one computer.[3] I must confess to being downright offended by the right unfriendly message that came forth the first time I shuffled operating systems.

V-SPOOL can make a big difference

I have complained (sometimes publicly) for a couple of years about an NEC 5530 Spinwriter (Centronics parallel version) that has groaned along under the control of WordStar like a sick duck at about 30 characters per second, even though I know very well that the printer should work much better than that. I have preferred the synergism of Lifeboat CP/M with WordStar on my favorite TRS-80 over the others and have accordingly generally used it for WordStar work.

It has not taken me long, however, in fiddling with V-SPOOL, to figure out that a switch to Pickles and Trout CP/M together with V-SPOOL, even in its minimal configuration with a 2k buffer, would clean up the sickly performance of my 5530 and make it purr like a well-oiled sewing machine (as indeed it should).

Conclusions

It is this writer's notion that V-SPOOL is a solid and credible product which is a positive addition to the field of utilities. I am inclined to recommend this product to anyone who thinks he needs a little better printer performance and who thinks that he may have a little memory to spare. Furthermore, since only those of us who use what is affectionately known as a "TRASH 80" get the "opportunity" to buy several operating systems so that we will have one that works for every (any?) occasion, I may even be harsh in griping about their clumsy software protection scheme.

[1]If you own a TI-810 you might be interested to know that TI has on the market a LQ upgrade kit for $750, that will give your TI-810 about six different type styles including two multi-pass letter-quality styles with a legible lower case (for a change). The kit consists of a new print head, a new paper advance motor, and another board to plug in the (hopefully) empty card slot in the back of the TI-810 card cage. It also has a 4k printer buffer built-in. I bought one some months ago and have not been disappointed.

[2]Tandy recently replaced the Model 16 with a Model 16b. We used to think that a computer was obsolete by the time that it had been on the market long enough so that it worked. Here we have a computer that was replaced before it worked.

[3]If you can't understand why I have four versions of CP/M 2.2 within arm's reach of my computer, consider this. My favorite spreadsheet (Perfect Calc) works only with Lifeboat CP/M. Lifeboat and WordStar work very well together (except as noted with a NEC 5530). Lifeboat does not support double-sided drives, however, which I have. ATON has a bankswitched version of CP/M which makes my CB-80 compiler run 25% faster due to its track buffering I/O, but causes one of my favorite editors (VEDIT) to explode in the bank switched mode but not the standard mode. ATON blows up WordStar in either mode, if you attempt to use memory mapping, but will allow WordStar to struggle along in the SOROC 120 emulation mode. Pickles and Trout does not emulate any standard terminal, and has some other minor, but disagreeable, personality traits.

Feature

CP+ And Simplifile — Programs To Operate The Operating Systems

by Charles E. Sherman

One of my current clients is a small agency in need of two or three computer work stations. The people who will use them are fairly typical for new office or business computer users, which is to say they are bright, conscientious, and apprehensive. These are non-technical folks who are struggling to hide the fact that they have strong mixed feelings about the long-heralded high technology which is about to appear suddenly on their desks. Part of their problem, of course, is simply the fear that they won't be able to handle the intricacies of the computer, that they haven't the aptitude for it, that it will be difficult. Part of my job, and presumably yours, is to help such people to make the transition as smoothly as possible, and this can best be done by making the system as accessible as possible. In a very common situation like this, having to teach CP/M as a first step is something I would often prefer to avoid. Wouldn't you?

Applications programs are getting easier to use, but unfortunately for sales people and consultants, CP/M is still as opaque to the new user as ever. Certainly, the basic tools and procedures can be demonstrated, written down, and mastered with a bit of effort. Anyone can handle the rudiments of STAT, PIP, COPY, REN, and DIR after a few hours' time, especially with talented teachers such as yourselves, and especially if you don't move too quickly into the refinements. But being confronted with the convoluted and cryptic logic of CP/M at the same time as having to learn their first word processor can make many new users long for their pencils, scratchpads, and typewriters. It certainly confirms their expectation that the computer is going to be a chore to learn. This is why I prefer to start off with a CP/M "front end," which is to say, a program which operates the operating system.

Front end programs for CP/M can do more than make life easier for the new user. They can also overcome some of CP/M's limitations. In my last column I discussed the need for better directories and file management features, which is one problem that a good front end program can solve. The featured program that month was Trakmaster, which was very well-conceived but not well implemented. This month's feature program, Simplifile, is less ambitious, but accomplishes everything it sets out to do and works very well indeed.

Simplifile

Simplifile is the first offering of Durant Software, a start-up company in Berkeley, CA. As its name implies, this $100 program is designed to make the file-handling tasks of CP/M simple. That it does. Among other things, Simplifile gives you added ID information on files, allows chaining of files for multiple operations at a single stroke, and facilitates the backup routine. There's more, so keep reading. It installs itself very easily on most popular computers and terminals, and since the publisher will cheerfully assist those with off-beat gear, anyone running almost any CP/M can use it. For those terminals with half-intensity capability, Simplifile makes modest but tasteful use of it.

The 32K program SF.COM is PIPed onto each diskette from a master copy. Each time it is invoked, Simplifile comes up with the screen shown in Figure 1. The cursor appears in the top three rows, the Disk Area, which contains disk status information, a running total on space left, and disk commands. The first time SF is invoked on a disk, you must give the disk a six character ID. Thereafter, each time SF is called from CP/M, it queries for the latest date which you can either give or ignore by hitting RETURN.

Disk commands are executed by positioning the cursor over the command and hitting RETURN. For added convenience, you can position the cursor on any command simply by typing its first letter. For example, to change the current drive (and the files listed in the File Area below) simply move the cursor to "Current:" and hit RETURN and B to log onto drive B. Similarly, to re-sort the directory listing according to filetype instead of filename, simply type "S" which moves the cursor to "Sort," then hit RETURN; the screen flashes sort options, and you type "T" to select a sort according to filetype. It's much faster than it takes to read about it.

The rest of the screen below the Disk Area is the File Area. You move to and from the two areas with the ESC key. The Help command or "?" will display the commands as shown in Figure 2. The disk commands in the left column explain the commands in the Disk Area at the top of the screen, and the right column explains the various file commands.

The File Area contains information about the files on the current drive. If your file list overflows the screen, additional screens of files can be viewed by hitting **N** (next screen), and you can move back by hitting **P** (previous screen). **T** moves the line the cursor is on to the top of the screen. The list of files can be sorted according to file name, file type, or size by using the Sort command at the top of the screen. The cursor is moved up and down through the file lines by use of the cursor arrows, and action may be taken on any file by putting the cursor on that line.

When the cursor is on a file, you can manually enter the date and a description of up to 42 characters to help you identify the contents of the file. If the date being entered is the same as the current date shown in the Disk Area, simply enter " = " and hit RETURN. You can also take action upon that particular file by using the file commands. Thus, you can **V**iew, **C**opy, **E**rase, **B**ackup, **L**ist, or **R**ename any file simply by entering a

one character command and hitting RETURN. Nice! With the **U**ser command, you can copy any file into any user area. Even nicer, you can copy, backup, or list any *group* of files simply by marking each selected file with "M" and then using the "Mult" command at the top of the screen.

Viewing with Simplifile is much nicer than using the CP/M TYPE utility, because the file is thrown up a screen at a time, and you can go forward or backward through the file with the N and P keys. You can also view .COM and other hex files, beautifully set out with addresses, numbered columns, and a column of ASCII equivalents on the right. Unfortunately, you cannot read more than the first 16 to 18 Kbytes of any file. It just quits. This is because the authors conceived of this feature merely as a utility to allow you to positively identify a file, whereas I would tend to use it as a means of quickly getting to some bit of information without loading the file with my word processor. Fortunately, not too many of my files are over the limit, and there exists the possibility that the authors will eliminate this limitation in a future enhancement.

Any program can be executed by pointing at it with the cursor and hitting "X". You will be queried for parameters (arguments) which you can enter if they are needed. Likewise, if you e**X**ecute any non-COM file, you will be queried for the name of the program which you wish to act upon. Thus, you can "X" EDIT.COM and enter the name of the text file, or "X" the text file and enter EDIT. It works either way. Simplifile comes with a handy 6K utility called FINETUNE.COM which, among other things, accomplishes two very important tasks. First, you can automate up to five of the most commonly used .COM programs — e.g., your word processor, spread sheet, and data base — by listing them under PROGLIST (program list). Then, whenever you e**X**ecute a data or text file, the names of the .COM files you listed will come to the screen with the cursor on the first one. Simply move the cursor to whichever .COM file you choose to have act upon the selected data file and hit RETURN. If you desire one which is not listed, put the cursor on "Other," hit RETURN and enter any .COM file you like. This is much more convenient than always having to type all the names in. Second, and here comes the very best part, when you use the PROGLIST utility, every time you e**X**ecute a file it can be made to automatically update with the currently listed date. I love it! This makes it very easy to go through a large directory and identify and backup all recently modified files all in one group.

This version, 1.2, works beautifully without bugs or errors. However, the error trapping needs work. When attempting to rename an R/O file, I was thrown out of SF and back to CP/M by the BDOS error, and the same thing happens if you try to access a drive which is not ready. This should be fixed, and a function should be added to enable one to change file status from SF. There are very few limitations, and it must be noted that Durant Software is not only friendly and helpful, they are also eagerly collecting suggestions for improvements from users and planning an eventual enhancement. For example, an important prospective feature will be the ability to format and set up new disks in a single stroke. It would also be nice to be able to hook up with an on-board clock for date stamps, but at present this is only possible on IBM-PC, and there are no plans for expanding in this direction.

Simplifile does not provide any facility or information for making it autostart, but anyone who knows CP/M should have no problem setting this up. One nice feature, which is undocumented, is that when invoking SF you can add parameters to name the logged-on drive and the sort order for the file listing. For example, **SF B T** will come up logged onto drive B: with files sorted according to file type. I use this all the time, but would prefer to be able to permanently alter the default options.

In sum, Simplifile is an excellent utility which offers advantages to any user, and especially to novices. The file description feature eliminates confusion about what is in your old files, and this will be a real advantage to any users who have large directories, or who accumulate disks over a period of time, or where various people put files into the same system. Any user will appreciate the way this program facilitates the backup routine. Consultants and salespeople will especially benefit from the way Simplifile makes the computer more accessible and attractive to customers and clients.

Sexier screens with CP+

CP+ is a $150 front end program from Taurus Software in Lafayette, CA. It organizes CP/M functions into plain English menus and adds extra features like password, print queueing, file chaining for copy/backup operations, and more. The program packaging is extremely slick, as are the promotional materials. Every effort is being made to attract and support computer sales outlets. If the marketing is as powerful as the packaging, they should be able to overcome the fact that the product is their weakest link, and sell it like crazy anyway.

The sexy packaging continues into the screen operations of the program itself, and this is the main, almost the only, reason I would recommend the product. I really like the way the screens work, and there seem to be dozens of them! A single keystroke sets them in motion, and they move like animation, taking full advantage of highlighting. The screen action is the program's strong point, and should be extremely useful for sales to first-time users. Whatever system you are showing will look great — just the way computer screens in the movies look — and the customers will be just itching to get one home and play with it. That's just fine. And while playing with it, they can slowly transfer their attention to applications programs, and eventually, like the ideal socialist state, CP+ can just fade away. Or maybe not so slowly.

As far as actually using CP+ goes, it will make you grind your teeth in frustration. This is a prime example of a program which is trapped by its own relentless and rigid logic. It is absolutely clear, yet horrible to use. CP/M operations are meticulously broken down into options, subcategories of options, and sub-subcategories. Each has its own screen, and that's the problem. Once the pleasure of watching all those screens in motion fades, you realize that you spend a lot of time waiting. For example, the starting screen comes up, then you hit ESC to get to the Quick-Screen lineup of opera-

tions. Let's call it QS for short. You can select any one of 18 possible actions by entering its number then hitting RETURN. Each action has its own menu and sub-menu screens which you must navigate through. Then when you accomplish that task, you go back to the QS by hitting ESC and start the next operation.

The third or fourth time I tried to copy one file from A to B, then check the B directory to make sure it got there, it took me six minutes and an unbelievable 30 keystrokes, *not* counting typing the name of the file, but by then I was getting good at it. The first time or two took much longer. Here's what you have to do:

1. ESC to the QS for the list of options.

2. Select 6 (Directory). Wait for screen to write, wait to fetch directory.

3. Advance through directory looking for desired file for transfer. CP+ makes you use "+" to advance through directory screens, a very awkward shifted key and a poor choice for a command used so frequently. Directory screens give filename and size only, and display only 20 files per screen. You cannot select files or sort the directory, you just have to wade through it. Ooops! A confusion arises. I see that I have both XD.COM and XDIR.COM, two similar but very different directory utilities. Which is which?

4. ESC to the QS.

5. Select 5 (File Description Catalogue). A nice idea; you can enter descriptive data in the catalogue next to any filename in it. Wait for screen to write, wait to fetch directory. Use "+" to advance through the screens. Each catalogue screen holds only six files, so I have to advance to the 5th screen to see both of my file descriptions. Aha! It's XD.COM that I want.

6. ESC to the QS.

7. Select 14 (Copy Some Files). Wait for screen to write. Select A (Add Files To List). Cursor moves to list area, enter filename XD.COM. Hit RETURN to accept it, hit RETURN to accept same name on copy-to side, hit B to designate copy-to drive, hit RETURN to indicate done with list. Select X to execute the copy routine, hit Y to say "Yes, do this line." At last, it's done.

8. ESC to the QS. Now I want to check the directory on B to see if I did it right and got the file copied over.

9. Select 6 (Disk Directory). Wait for screen to write. Wait for it to fetch the A drive directory. Select C (Change Drive). Hit B to change to the B drive. Wait for it to fetch the B directory. Use "+" to advance through screens until I see XD.COM. It worked! Select C (Change Drive). Hit A to change to the A drive. Wait for it to fetch the directory.

10. ESC to the QS and get ready to start the next job.

What a horrible way to work! If I could spare any hair, I'd have pulled some out by this time. I can only assume (and hope) that someone expert in the use of this program could do it faster and with fewer steps, but I think this will give you the general idea. By contrast, using Simplifile to copy or transfer a file is almost as fast and easy as using CP/M, even for a veteran hacker. Because of its rigid and intricate logic, CP+ is nearly as difficult to learn as the fundamentals of CP/M, and not nearly so worthwhile. I have similar reactions to their documentation which, despite its gorgeous and artistic appearance, is actually quite obtuse.

The viewing function of CP+ is alarmingly flawed. Unlike Simplifile, this program is extremely fussy about what it will view. If it sees any non-ASCII characters it throws out yet another of its graphical displays and flashes wildly that this is a non-text file. Unfortunately, quite a lot of programs, including word-processors like Spellbinder and Palantir, pack in a few formatting characters, and this gives CP+ the fits. So whereas Simplifile will allow you to view more than is possible with CP/M's TYPE command, CP+ is far more restricted. And even when you can display a file, you can only view it in the forward direction.

Why bother to spend all this time panning this product? Well, for one thing, you will probably be seeing ads for this one, but there's a much bigger reason. The major problem with CP+ is so extreme that it makes an excellent example of an approach to programming and problem solving that every programmer should learn to avoid, and something that every consultant should beware of in any program selected for clients. It is the same problem which afflicts Select and various others. No program should ever allow rigid adherence to its own logic to usurp fluidity and flexibility of use. One should never sacrifice the user's convenience on that particular altar. Being clear and logical is not a program's highest goal, but merely a contributing factor to its ergonomic success. If having to wade through interminably branching menu choices is "user friendly" then beware those programs which will kill you with kindness.

In sum, CP+ is fun to watch but no fun to use, so I could recommend it as a good computer sales device, but for no other reason. My sincerest apologies to the fine people at CP+. It is never fun being hard on a product, as I am increasingly aware of the high degree of effort and commitment required to get a baby like this to market. Sweat, dedication, and professional talent is reflected in the extremely high quality of packaging and in the assemblage of fine marketing materials. I just wish I could like their program better.

DURANT Software
2532 Durant Avenue, No.250
Berkeley, CA 94704

TAURUS Software
3685 Mt. Diablo Boulevard, No.251
Lafayette, CA 94549

(continued on page 10)

Feature

A Review Of MemoPlan

by Marilyn Harper

MemoPlan is a highly specialized text editor which can accommodate from one to seven concurrent or "active" documents. The documents can be worked with in round-robin fashion, and portions of two of them can be displayed on the screen at the same time. Interactions between the two screen windows are so well implemented that I felt this was the single most impressive feature of the product.

I won't attempt to rate MemoPlan exhaustively on subtle points, but just present here my general evaluation, based on about 12 hours of use.

User Friendliness:	Excellent
Safety:	Excellent
Editing Features:	Very Good
Printing Features:	Fair (but not elaborately tested)
Documentation:	Excellent
Installation:	Good (when user-installed)

Now let's look at some of the details....

MemoPlan, as its name implies, appears to be geared toward business office writing tasks. Usage suggested in the manual includes putting a list of things to write in one window, while using the other to actually do the writing and editing involved. Office memos would tend to be short text files, but this editor could handle a single file of up to 248K bytes, if you should wish to use it for such massive works.

The active documents are "part of" MemoPlan, until intentionally discarded by the user. This semi-permanent storage is handled by what's called the "swap file." As Memo runs, it periodically and automatically writes the currently-selected document to its segment of the swap file. You may also want to explicitly write your documents to, and read them in as, ordinary ASCII disk files.

I found it a little disconcerting at first that each time I brought up Memo, it showed me the last screen of the last document I had been working with when I had last exited. However, the target group for this editor may very likely regard that as one of its most helpful features.

MP should not be loaded with high-bit-set document files, but will accept plain text produced by other editors (e.g., WordStar non-document files). I occasionally saw a few inexplicable reverse video characters crop up on the Zenith screen when I read in plain ASCII files, but this didn't happen often enough to be a serious problem.

The only difficulties of any note that I encountered with MP arose while running its installation program, CONFIG. I didn't regard this as a serious flaw; I did get the editor up and running. The manual implies that most users will have the program installed by their dealers. Still, I'd like to mention the problems I found.

I was sent a review copy of MemoPlan for 8-bit CP/M micros. Two very different computer systems were available to me, and I decided to install MP on both of them. (Note: this would be a violation of Chang Labs end user license agreement, but was done for the purpose of review and evaluation only).

The first computer was a Radio Shack Model II with 128K of bank-switched memory, four 8" floppy drives, and the RS Daisy Wheel II printer. I started out by putting the MP files on an Aton CP/M disk, but could never get the install program to run at all under Aton. I gave up and installed MemoPlan on a Lifeboat CP/M disk. The Aton problem was just with the installation program; I could boot Aton in drive A and run my already-installed Lifeboat MemoPlan disk in B.

Printer installation appeared to be easy; the Daisy II was on the menu (but see comments in the discussion of Printing Features). I installed the terminal as ADM-31, as one would do, for example, for WordStar under Lifeboat CP/M. Eager to see MP, I brought it up and was soon entering text. To my dismay, my text was appearing in reverse video. I soon exited from Memo, and discovered that the terminal had been left set for reverse video, which was intolerable. I went back and installed for the ADM-3A terminal, thus dispensing with reverse video altogether.

I did go back once more to install for the TRS-80, when I noticed that the cursor arrow keys were not being supported, as the manual implied they would. I had glanced at the Advanced Configuration Menu, which had a 'Define keyboard usage' option. I hoped I could enable the arrow keys by remapping them to Memo's cursor controls.

I never got anywhere with this; the CONFIG program always blew up with the message "Fatal error: can't open the overlay file."

I also installed MemoPlan on a Zenith-89 computer with 64K memory, 1 on-board 5 1/4" floppy drive, a 6MB Corvus hard disk, and the Zenith Z-25AA line printer. In this case, terminal selection was easy; Z-19 was on the menu. The printer wasn't — it's an interesting printer, but few people would choose it for letter-quality applications.

For printers not on the menu, CONFIG offers the choices 'Plain' and 'Vanilla', which you might want to try just to see if one works. I decided to try for a custom installation. This begins when you admit to CONFIG that your printer isn't listed; follows a fair number of questions about what your printer can do, and *voilà*, in this case I soon had the printer installed. Or partially installed — the custom installation worked fine for printing from within Memo, but see comments later in this review for Printing Features....

The Z-19 screen installation is undoubtedly how MP is intended to look. There is a reasonable use of reverse video for menus and status line messages. ►

The Zenith arrow keys weren't enabled either, and I tried to use the 'Define keyboard' option, but with the same non-results as on the Radio Shack.

User interface

The MemoPlan user's manual is outstanding. It is short, easy to understand and to use, and attractively formatted. Command keystrokes are printed in huge bold letters beside the text which describes their usage. Both a Table of Contents and Index are present.

MemoPlan's menus are attractively presented on the screen. Selections have well-chosen mnemonic names. The small first menu is on the screen at all times, unless turned off by the user. Two additional menus may be called up, giving progressively less common selections. None of the menus overwhelms the user with too much information.

At the bottom of the screen, a status line tells you the following:

- Name of current document. Called "NAME-ME" if never written to a disk file; but changes to user-selected file name afterwards.
- Percentage position of the cursor in the document: 0% is the top of the file, 100% is the bottom.
- Current text mode; insert mode by default.
- Forward-Reverse direction toggle setting.
- "Swapping" message appears when Memo automatically updates the current document in the swap file.

MP includes a Help command in its main menu. It gives information about a specific key, or you may type in a word which you want to know something about. I didn't catch on immediately that I needed to use all lower-case for help with command words. But at least I was told that "Help about 'PRINT' not available. . . ."

One of the first things I did with Memo on the hard disk was to reset its document parameters to be as large as possible. It had arrived on the disk set for five documents and a maximum of 16K of text storage in the swap file. The user can set the number of documents from one to seven, with a maximum storage of 248K. You might never need that much space for seven documents, but it's nice to know you can have it if you want it.

This is done with a versatile utility called RECOVER. Re-sizing does destroy the current documents in the swap file, but they can be stashed as plain ASCII disk files, and read back in after the operation. RECOVER is impressively bullet-proof; asking it to run with totally inappropriate parameters results in a message on correct usage, and a firm but friendly return to the CP/M prompt.

The main function of RECOVER is to restore your swap file if it gets glitched up by a power failure, accidental hardware reset while swap file is open, or whatever. I tested this by doing a reset on the TRS-80, then running Memo. A message tells you the swap file is invalid, and to run RECOVER. This takes just seconds with the 16K swap file, and you're ready to go again.

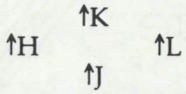
I also found RECOVER to be useful for keeping track of how large each document in the swap file had become, and how much room was left of the total. All this is given in a sort of status list on the screen when you run RECOVER. Also reported is the number of killed documents which have been cleared out by the user. (The usefulness of knowing this isn't obvious to me.)

Memo will inform you when the swap file is full. You might then want to clear out an old document, or write one or more out to disk and free up their space. However, the manual instructions are very good for the use of RECOVER to increase the size of the swap file.

Editing features

Overall I found MP's editing commands to be very good. My only complaint was a personal one — I like to have the option of reassigning the cursor controls. And perhaps this could be done with a working version of the 'Define keyboard usage' part of the install program.

I didn't care much for MP's cursor control keys. The manual depicts them in a typical-looking diamond pattern, which is a bit misleading, since the four keys are in the same row. A real diamond would be much easier to use. . . .



(Ed. Note: This configuration is default on several popular terminals.)

Memo has four different editing modes. I didn't find their names to be especially helpful, except for 'Overwrite'. I thought that 'Fill' should have simply been called "Insert Mode." 'Justify' does a sort of word-wrap at the right margin. 'Normal' requires a hard carriage return to end a line before you reach Column 80.

Memo uses the destructive backspace or Rubout (ASCII 127) as its only command to delete a single character. This will pose something of a problem on keyboards which lack this key. The TRS-80 Model II has no labeled Rubout key, but sends decimal 127 via CTRL-'underline-and-hyphen' key. Potential purchasers of MemoPlan who lack the Rubout key should check their hardware manuals to be sure they have some way to execute this function. I found that I could insert blanks around a character I wanted to delete, and use MP's 'Delete Word' command to get rid of it, but this was tedious.

A feature of MP which is new to me is its Forward-Reverse direction toggle. Its current setting is always given at the bottom of the screen. It usually reflects the last cursor-control key used, but it can be explicitly reset very simply with ↑F for Forward, ↑R for Reverse.

Many of the MP commands have a dual meaning, depending on this toggle. Examples: in Forward mode, ↑W moves the cursor ahead 1 'word', while Reverse mode ↑W moves back one "word." (I say "word" because the action is a little strange: Reverse Word puts the cursor on the first character of a word, but Forward Word puts it at the blank in front of the first character).

Forward ↑B puts the cursor at the end of the file, while Reverse ↑B moves it back to the beginning of the file. There are four additional, intermediate degrees of cursor movement which are used this way. Indeed, it seemed that the only feature which didn't utilize the direction toggle was the ↑G to Get next document. You can jump directly from file #3 to file #4, but not vice versa in One Window mode. You have to cycle through

all of the intervening documents, which takes just seconds to do.

Memo's Two Window Mode is elegantly executed and very convenient to use. When two windows are selected, the screen is split horizontally, with part of one file displayed at the top, part of the other at the bottom. To test the use of this feature, I loaded the top window with a 4K ASCII file, which was the Table of Contents for a manual I had been writing. In the bottom window I loaded 50K worth of Chapter 3. The idea was to scroll through both (at different rates), checking the index against the text. I was satisfied with the performance, but wouldn't want to do something like that routinely.

The following Two-Window operations are handled very smoothly: switching the cursor between the two files, scrolling the two files with use of the Forward-Reverse toggle, moving blocks of text from one file to the other, allowing the user to change the size of one window in relation to the other. Four different screen allocations are possible, ranging from seven lines for the top file and 10 for the bottom, to 12 at the top and nine at the bottom with the menus turned off.

Memoplan has a convenient Undelete command which restores the last deleted item — word, sentence, block, etc. — at the current cursor position. You couldn't restore an entire document which you had discarded by using Undelete, but the Clear command is protected by a confirm prompt.

Undelete is the basis for MP's move and copy functions: an item is deleted in one place and undeleted somewhere else. Moving text between two windows is handled the same way. You delete the item in one document, undelete it in the same spot if necessary, switch the cursor to the other document, and do an undelete there. It's faster to execute than to describe.

Setting markers for blocks of text works well but isn't visually spectacular, *à la* the highlighted blocks of WordStar. Setting just one marker results in an electronic "bookmark" that holds your place in the document until the next time you want to locate it quickly.

Find and replace functions are well-implemented. MP's Search is the only one I've personally encounterd which will match search string "capital" with found strings 'capital', 'Capital', and 'CAPITAL'.

The Query Replace has an interesting option called 'Try it', in which the substitution is made on the screen, and then you decide Yes: leave it changed, or No: go back to the original version. The search then continues, so you don't get to see the results of your 'No' response, reverting back to the original, which I thought would have been more satisfying.

Printing features

I didn't spend a great deal of time investigating MemoPlan's print formatting capabilities, because I didn't like the way printing is handled. Print parameters, or options, are entered as a string of switches typed at the keyboard. Until you had these memorized, you would have to look them up in the manual.

Compared to WordStar, where the user responds to a series of screen prompts, I found this distinctly inconvenient. So usually I just hit Return to get all defaults, so I wouldn't have to fool around with the "option list."

The print procedure also seemed unnecessarily slow. Two files, MEMO.F$$ and MEMO.P$$, are written to the disk, and there is a final Confirm-Print prompt, before your document ever goes to the printer.

The MemoPlan package includes a separate program called MEMOP .COM which handles all of the actual printing. The manual describes direct invocation of MEMOP from CP/M to print ASCII files. It sounded slicker than just using TYPE ↑P, since you could get title headings on each page, set tab expansion, and a few other goodies.

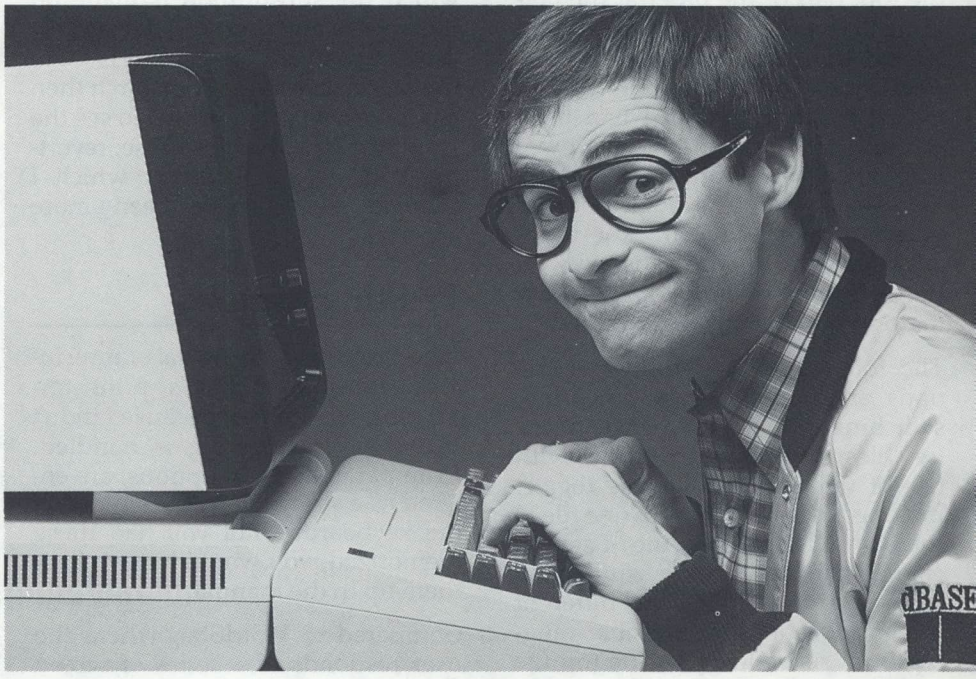
But MEMOP had some annoying problems when I ran it as a stand-alone utility, as opposed to invoking it with ↑XP from the Memo menu. I tried direct MEMOP for the first time on the Z-89, and absolutely nothing went to the printer. I could send my ASCII file to the Z-25AA with TYPE ↑P, or with Memo's ↑XP, but not with MEMOP filename.

I went to the TRS-80 system to test direct MEMOP there. The Daisy II printed garble, with missing lines, incomplete lines, and extra blank lines in the printed file. A friend commented that stand-alone MEMOP didn't seem to be driving the Daisy II correctly, and suggested re-installing for the 'Plain' printer. This solved the immediate problem, namely of cleaning up the output from directly run MEMOP.

I went back to the Z-89, re-configured its printer installation from Z-25AA to 'Plain', and found that direct MEMOP then worked in that case too. I did check to be sure that printing with ↑XP from within the Memo editor itself still worked, but then didn't pursue the matter any further.

I never felt that the printing facilities of MemoPlan came anywhere close to the high standard reached by its screen operations. After the direct MEMOP episode, I began to feel even more strongly that MemoPlan's printing features still need a lot of tweaking up. After all, "memos" are ultimately destined to become text on paper, to be distributed around the office. . . . ■

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

Feature

Computers And Organizations: Technology And Administrative Groups, Part One

by Jon Wettingfeld

The pundit's perils

It is with trepidation and a wary eye towards the long list of predictive failures cited in a recent *Lifelines* editorial that we begin a two-part series on long-range perspectives of information technology. Certainly the hubris of zealous marketing people, with their "sound and fury," is regrettable when it misleads the end user, causing him or her to opt for a system that isn't right for his or her needs. And if so many "false prophecies" can spring from a brief five- or ten-year span in the (rapidly evolving) microcomputer industry, how can anyone hope to predict long-range trends in such a volatile area?

Happily, this article avoids the slippery slope of fine-tuned predictive posturing. It looks not so much at historic detail, market successes and failures of operating systems, etc., as at the more general questions of what impact new technologies can be expected to have on organizations. A related question, what are some properties of organizations in general, might also be asked. Answers to these questions may be found by 1) looking at historical examples, and 2) seeing if changes in organizations can be related to changes in technologies on the basis of a dependent or independent variable relationship (i.e. the "chicken and the egg" question).

Answering such questions may, in turn, allow some coherent crystal-gazing. (We should note that by "organizations" we mean both business organizations and political administrative groups.)

Looking Backward: The first 5500 years

At first glance, the answer to the first question,"what has technology done for organizations" seems self-evident. Technology and civilization have grown up side by side. Right? Unfortunately, the answer seems more ambiguous than that.

For instance, it is commonplace in computer literature to give a brief history of technology from the invention of writing, all the way through to computer and word processor, and to view the technology as "causing" change. Looking at history as a rough sequence of inventions is not entirely inaccurate, but doing this overlooks the big picture of how complicated the changes that occurred were.

Many facts contradict, for example, the simple technological determinist view of the impact of writing on history. Some examples: 1) "Civilization" itself arose more than 3500 years B.C. in Mesopotamia — before writing was developed. 2) The technology of information storage that was utilized in these early states (a simple system of clay counters, seals and containers) may have existed long before the emergence of these first full-time, specialized, multi-tiered administrative organizations. (See Denise Schmandt-Besserat in Recommended Readings, below.) 3) It is a matter of record that another such administrative group, the largest in the new world, was almost literally held together by mere "pieces of string," by which I mean the Inca Empire, occupying, at its peak, the entire west coast of South America, and incorporating 80 provinces or states, which kept records on knotted pieces of rope. These people ran a government very efficiently, conducting censuses, allocating croplands, and of course, collecting taxes. All was done without writing, utilizing only the knotted Quipu records of vital state information.

Moving into the present, and demonstrating how uncertain the technological/organizational connection can be, we can cite Vincent E. Giuliano, a senior member of the information systems section of Arthur D. Little Inc. Writing about the spaghetti-like picture that presents itself when one looks for the relationship between computers and businesses, he suggests that the transforming effects of computer and communication technology on office work cause "A complex set of feedback loops. . ." linking economic and social changes, new developments in information technology, the widespread application of the new technology, and the introduction of the new office organization the technology makes possible" (see Recommended Readings). (However, Giuliano does feel that such technology has an important effect on office work organization.)

Thus the "time-line" school of "technological change propelling organizational growth" may well oversimplify our picture of both the past and present. This is not to minimize the impact of technology *per se*. For instance, we know that the past century has seen the service sector of the economy overtake and surpass the agricultural and industrial sectors in terms of both the size of its labor force and the dollars generated within it. This shift in the peopling of industries has been due largely to the automating of the "goods producing" sectors, as well as the internationalization of the economy. Still it certainly seems conceivable that some expansion could have taken place without new technologies in the *administrative* sector. Global economic neworks already existed before the beginning of the nineteenth century!

Given the above uncertainties of the connections between technology and social systems, but assuming we could predict such things, what sort of impact might we expect the new innovations to have on modern social organization?

Decisions, Decisions: Some views of complex organizations, their history, functions and growth

Looking at the development of administrative organizations can suggest a few possibilities. From a his- ►

torical perspective, organizations with full-time specialized administrators probably developed in connection with their superior ability to monitor and manage resources. This enhanced ability may be due to a number of factors. Some writers have pointed out that central integration of production of a variety of goods or services creates increased efficiency. And others have suggested that efficient activity coordination may be the basis for the hierarchical structure of decision-making organizations. Along these lines, anthropologist Gregory Johnson, citing group dynamics studies made in settings ranging from small social groups to large corporations, has suggested that there is a maximum number of subordinate units in an organization that an administrator can monitor efficiently. Exceeding this maximum beyond a specifiable range causes decision performance to deteriorate increasingly rapidly. Thus more efficient management of resources and information would be expected to accompany vertical and horizontal specialization of administrative tiers. This specialization, in turn, keeps the maximum number of "channels" monitored to within the limits of efficient performance by the individual.

Whatever the reason for their success, there can be little doubt that once the first complex decision making organizations developed, they out-competed (or were more frequently selected than) the (presumably) less-efficient ones. A factor for the success of such organizations may be a tendency for fast growth because of a potential to efficiently incorporate groups into a larger organization than they have got, rapidly. Thus one worker has noted an exponential decrease in the *number* of autonomous political units in the world over time. Johnson has suggested that this trend may be due to the fact that the linear increase in the number of levels of a hierarchy may generate an exponential increase in the potential system size — this within certain constraints of operation. Herbert H. Simon, an administrative and social scientist, has suggested a similar pattern of growth among hierarchical systems in general.

To sum up, when complex organizations first appeared and grew, growth was closely linked to their advantage in processing information, coordinating activities, and making decisions, and also to the intrinsic, readily-expandable nature of hierarchical systems.

The most important limits to organizational growth, according to some economic geographers and archaeologists, may be related on the one hand to increased transport costs of serving and administering an increasingly large area, and on the other hand, the problems of intra-organizational communications. Addressing the second point, Johnson cites Williamson's model of control loss in organizations. Williamson, writing in the 1967 (vol. 25) issue of the *Journal of Political Economy,* suggested that potential organizational control decreases with an increase in the number of organizational levels. This loss is due to information loss and distortion as the information travels between levels. Interestingly and importantly, improvements in data processing can, he says, minimize this effect. Thus a "stress point," a limit to organizational growth, may be removed with the development of proper technology. Here, then, is a point to consider when evaluating the impact of data technology.

It is interesting to speculate how both geographic (think of the new, long-distance communications technologies) and intra-organizational limits on growth might be minimized by the advent of multi-tasking, distributed data processing, on-line processing and video text systems. In fact, it is precisely the removal of hindrances to efficiency and growth to which some writers in the computer field address themselves.

Looking Forward: A big impact on small groups — and others

An example of this view may be seen in work by C. Roger Smolin. He has examined the impact of micros from the pragmatic standpoint of their uses in business. In his very delightful and readable "How to Buy the Right Small Business System," he describes almost exactly the type of logistical problems of information transfer to which Williamson refers. Noting that moving from the "orderly din of the production floor" to the office area leads one to the incongruous sight of billing, inventory and payroll systems being run via the channels of paperwork that are more characteristic of an earlier age, he suggests that these channels are imperfect indeed. Yet they are summarized and reported to management, who receives the reports in an "overjoyed" state. This, because they are a means to "symbolically represent" details "in the real world of the production floor, the suppliers, the marketplace," allowing management to "see the business" and make decisions.

According to Smolin, the impact that small computer systems will make on such organizations includes the improvement of this "picture." Part of the results of this improved information are not immediately perceptible; these include the "peace of mind" provided by a well-run business, and increases in productivity as well as enhanced opportunity because records of payments to vendors, payments from customers and inventory available to customers will all be more accurately represented. More important in terms of a perceptible "payoff," he suggests that small systems can be used as a TOOL FOR GROWTH. For example, once the accounts payable system is automated, there is little difference between having 20 customers and 200. It is here, he feels, that a switch to an automated system from a manual one is likely to have the most impact.

Giuliano, like Williamson, refers to improvements in information flow. He explicitly points to the reduction in information "float," the delay and uncertainty caused when information is inaccessibly caught in the mail, in the typing stage, or is misfiled. He also points out the potential for increased productivity because of elimination of redundant tasks like retyping and manual filing. And, he adds, better information flow leads to better decision making.

The picture that emerges is one of increased productivity, and a tremendous potential for growth, all "enabled" by better information networks. While, in the final analysis, "It needs no ghost to tell us this," it is hoped that a somewhat historical and theoretical perspective has provided some food for thought about organizational development, and the im-

pact technology may have on that development. While inputs of new technology need not be "revolutionary," it seems plausible that, given the nature of complex organizations, they may well approach that if they remove significant constraints on communication and growth.

Our next segment will look at the many changes going on in the workplace today as these new systems are introduced.

Recommended Readings

Giuliano, Vincent E. "The Mechanization of Office Work," *Scientific American*, September 1982, Vol. 247, Number 3.

Johnson, G.A. "Organizational Structure and Scalar Stress" in *Theory And Explanation In Archaeology*," Colin Renfrew, Michael Rowlands, Barbara Abbott Seagraves, editors. 1982, Academic Press, New York.

Simon, H.A. "The Organization of Complex Systems" in Pattee, H.H. (ed), *Hierarchy Theory: The Challenge of Complex Systems*," New York, George Braziller. 1973.

Schmandt-Besserat, Denise. "The Earliest Precursor of Writing," *Scientific American*, June 1978, p. 50, Vol. 708.

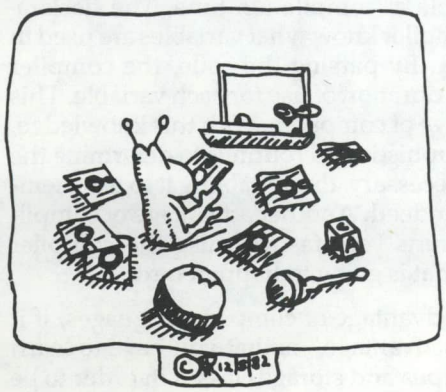
Oops!

In the May 1983 issue, p. 32, New Versions. . .

WordStar and MailMerge (CP/M-86) should be **3.3**; and dBASE-II 86 and PC should be **2.24**.

In the July 1983 issue, p. 33, Bugs. . .

The bug is for C-Food Smorgasbord **1.4**.



Feature PL/I From The Top Down — Chapter Two: Functions

By Bruce H. Hunter

In the last article, we discussed list and edited I/O, and we looked at both edited and list input/output techniques. We also looked at some loop forms, the call, procedures, blocks and the concept of scope of variables. I erroneously stated that we would be covering edited I/O in detail in this article, but in fact that will not occur for several installments. What will be examined in this article are declarations, data types, files and built-in functions.

This series of articles is rewritten from my book PL/I From The Top Down, *which is about PL/I Subset G, one of the subsets of the full set of PL/I. I deliberately take an informal approach in writing about the PL/I language, but if you are interested in pursuing the subject further, I think the best introductory text on the full set of PL/I is* PL/I Structured Programming, *by Joan K. Hughes, John Wiley & Sons, 1979, Second Edition. The compiler used for the writing of these articles is Digital Research's PLI-80.*

DECLARATIONS AND DATA TYPES

PL/I is a compiler language. Compiler languages are very different than interpreter languages, and now is a good time to discuss the differences between the two in detail.

Interpreter languages

Interpreter languages examine and execute each line of code before going on to the next line. Because only one line is executed at a time, an interpreter language has no way of knowing what is going to happen in the next line of the program. It never knows what is ahead. Line numbers or line labels are needed so if you need to use a "goto" the interpreter will know exactly where it should direct the program flow. Because interpreter languages have special symbols that automatically predefine data types and preset storage areas, the user of interpreter languages can be blissfully unaware of the significance of the various data types that exist in more sophisticated languages. For example, in interpreter BASIC a "string" is designated by a dollar sign after the variable (such as A$). That is all the interpreter BASIC user needs to know about using a string variable in programming. What is actually happening is invisible to the user — A$ tells the interpreter certain predefined things — the data is a string (type character) consisting of any number of characters up to a certain amount allowed (the amount depending on the version of interpreter BASIC), and the storage required for such is a predetermined default amount, the parameters of which also depend on the version of BASIC. Interpreter languages are handy for learning programming because you only need to know the "basics" to get started in programming. Also, changes to the code can be made easily. There is no compilation process required before the program can be run because interpreters only read one line of code at a time, so all you do is change the code, save it, type 'run', and you're in business. The disadvantages of interpreter languages are that they execute code very slowly and they do not use memory efficiently.

Compiler languages

Compilers are a lot different from interpreters. Compiler languages know what is ahead because they "parse" (read each line of the code from the beginning of the program all the way to the end) several times before the program is ready to run. To people used to interpreters, the compilation process can be a confusing experience. Let's look at what happens when code is compiled in a "typical" compiler. (Compilers are relatively standard in the way they compile code, but there are small individual differences from compiler to compiler.) Let's say you have just finished writing your source code, and now you are ready to compile it. The first step in the compilation process is accomplished by the preprocessor, which follows the instructions of the header file in your source code to "include" any header, defined or redefined functions, or constants. The next step is performed by the compiler proper which translates the code parsed by the preprocessor into assembly code. Then the optimizer optimizes the assembly code, making it more efficient. What results is what is called a "relocatable module," which is intermediate code. Finally the linker links the program proper to the specific library functions called by the program and to any other modules specified at linkage time to produce the final object code. Only after all of this is the program ready to "run." And when you change a line of code in a compiler language, you have to go through the whole process again before you can run the program! That's one of the only disadvantages of compiled code.

Compiled languages do not have symbols like the dollar sign to designate strings. It is up to the programmer to let the compiler know everything it needs to know about each variable. The place to do this in PL/I is in the "declaration" section of the code. Here each variable is "declared" (defined) as to its data type and storage class, because in order to determine the storage needed, the PL/I compiler needs to know the data type and storage class of each variable at compilation time. The declarations also let the compiler know what variables are used in each program block. By parsing the code, the compiler also determines the duration of use for each variable. This is a definite advantage of compilers. With this knowledge, the compiler uses sophisticated routines to determine the minimum storage necessary, thus enabling it to use memory very efficiently indeed. Another advantage of compiled code is that it runs very fast because the compiler knows everything that is going to happen next.

The only other disadvantage of compiler languages, if it can be called a "disadvantage," is that you have to learn about all the data types and storage classes in order to be

able to use them. In this article we are going to take a look at some of the data types in PL/I.

Data types in PL/I

PL/I Subset G has a number of data types, such as fixed binary, float binary (single precision and double precision), fixed decimal, float decimal (not implemented in Digital's PL/I-80), character, pointer, label, entry, file and bit. This is a lot to bite off in one chew, so we're only going to cover a few of them in this article — fixed binary, float binary, fixed decimal, character and file.

Data types for numeric data

Let's look at the data types for use with numeric data first because they are pretty straightforward:

Fixed Binary
Fixed binary is used for integers only. The range is from 0 to ±32767. In systems programs you can get by just fine with this data type, but for most programmatic applications you need to allow for a larger range of numbers. It's interesting to note that until recently, the 8-bit C compilers available were integer-only versions, and many sophisticated utilities were programmed in them (such as MicroShell which was initially programmed in BDS C). Incidentally, the term "binary" signifies the internal representation of a number with this data type. What you see on the screen is a decimal representation of the binary number stored internally.

Float Binary
Float binary is for very large or very small numbers. PL/I offers you a significant amount of programming power in choosing precisely what you want in terms of numbers to work with. You can opt for single precision or double precision in type float. As a reference, most other languages have double precision or single precision, but few offer you both. For example, CBASIC is double precision only. C has double precision and single precision, but computations are done only in double precision.

Single Precision
In single precision the type float can have a precision of up to 2 raised to the 24th (1677216), with an exponent of as much as 38. (The number of significant decimal digits is about 6.)

Double Precision
In double precision, type float can have a precision of up to 2 raised to the 53rd (about 15 decimal digits) with an exponent of as much as 308! That is precise! Double precision is storage intensive, of course, as it uses twice as many registers as single precision. Double precision also slows down execution time significantly. Nothing comes free!

Not only can you specify double precision or single precision, you can specify exactly what parameters you want. For example, the way you express the exact precision of type float is with a number in parentheses after the word "float," like 'float (15)'. The '15' signifies 2 to the 15th power, which falls within the single precision range. As far as I know, PL/I is unique among languages in offering you this kind of choice in determining the precision you need because other languages have default amounts of precision only. If you type float and no precision number, PL/I defaults to single precision (24). If you are in doubt as to the precision you need, just pick up a handy calculator and input the number of the precision as the exponent of 2, like y to the x with y as 2 and x as the number in question. If x is 15, the largest number possible is 2 to the 15th, or 32768.

Fixed Decimal
Fixed decimal has its precision in actual decimal places. Fixed decimal is like CBASIC's or Pascal MT-Plus's BCD (binary coded decimal). It is for exact representation of numbers, like dollars and cents, where truncation and approximation errors can't be tolerated. The precision is 15 digits maximum for this data type. Graphically that looks like $9,999,999,999,999.99, which means you can carry a number equal to or less than a penny short of ten trillion (with change) without getting in trouble!

Declarations with "numeric" data types

Declarations precede each program block in which the variables are to be used, and they take the following form:

```
dcl
  variable_name data_type
  (variable1, variable2, variable3) data_type;
```

The statement 'dcl' (which is short for "declare") tells the preprocessor section of the compiler that some declarations are coming up. The variable name (or group of names) follows with the data type at the end of the line.

Multiple declarations are not only permissible, they are common and they are also a handy way to demonstrate the way numeric data types are declared:

```
dcl
  i fixed;
  hyp float(24);
  (a, b, c) float(30);
  amt fixed decimal(7,2);
```

In the above example, the variable 'i' is type fixed binary. The variable 'hyp' is type float binary, single precision. The variables 'a', 'b' and 'c' are type float binary, double precision. The variable 'amt' is type fixed decimal with seven digits using two decimal places.

Data type character and declarations with data type character

Now let's look at the data type 'character', which is used for alpha data. Strings are alpha data, for example. It's an interesting fact that not all languages treat strings the same way. C and Pascal can only deal with one character at a time, so strings are dealt with as character arrays (arrays of type character). PL/I, like BASIC, doesn't have to deal with only one character at a time, so a group of characters doesn't have to be dealt with as an array of characters. Strings are specific entities in PL/I, and they are treated as such. In PL/I, when you want to declare a string, you not only have to specify that you are dealing with type character, you have to tell the compiler the length of the string. The length of a string must be declared:

```
dcl          buffer character (254);
```

In the above example, the string 'buffer' is 254 characters ►

long, no more and no less. If you forget to specify the length of the string, the default length is 1. PL/I is very literal. When you specify character, it gives you one character!

There is a 'varying' attribute for type character that allows the lengths of strings to vary as long as a maximum length is specified, as in the following example:

```
dcl
  name char (32) var;
```

In the above example, the string 'name' is type character and may be anywhere up to 32 characters long.

Some closing remarks on data types

To put PL/I in perspective, it has one more data type than C but twice as many data types as Pascal. Unlike Pascal and C, PL/I has no provision for creating your own data type. This does not significantly affect the programming power of the language, but I point it out as a matter of interest. There are many other data types in PL/I, but with a data type to handle character data and a data type to handle numeric data, the only other data type you need to write elementary programs in PL/I is the data type 'file'. We will come back to the other data types in PL/I as we go along, but right now let's go on to files. Files are not only another data type, they are a very important concept in PL/I, so they deserve a whole new heading.

Files

PL/I is called a "file-oriented" language. It not only deals with files in the ordinary sense, it treats just about everything as a file, including devices like the console CRT, the keyboard and the printer. This can be confusing at first, but this feature gives you a great deal of programming power. All I/O in PL/I is done by way of files. Everything with a 'put' or a 'get' is a file. Let's say you want to print the statement "All I/O is a file in PL/I." This is the way you do it. The first thing you have to do is declare a variable with the data type file, like this:

```
dcl
  line_printer file;
```

This declaration tells the compiler that the variable 'line_printer' is a file. Then a little later in the code the 'line_printer' file is opened, like this:

```
open file (line_printer) print linesize (80) title ('$lst');
```

PL/I wants its files declared (or defined) and then opened. Here 'line_printer' has been opened, and 'print' signifies a print file. The word 'linesize' signifies the option PL/I offers you of setting the line size (the length of the line before an automatic carriage-return line feed is put in). Here the line size is set to 80 characters. The 'title' or name of the file 'line_printer' is '$lst'. Here the title of our file is also the name of the listing device, CP/M's logical device (LST:) for the printer. Now at last you can have the line printer print the statement "All I/O is a file in PL/I" with this program line:

```
put file (line_printer) list ('All I/O is a file in PL/I.');
```

This will put to the 'line_printer' file our little statement "All I/O is a file in PL/I."

Right now some of you are wondering about all those "puts" and "gets" we did before. No one said a darned thing about files then! That's because PL/I has a couple of default files: SYSIN (system input), and SYSPRINT (system print). They're automatically open at all times, and a 'put' or a 'get' without a file assignment defaults to the CP/M default device (CON:) which is the console. Whether you do it implicitly or explicitly, the result is the same. The following two lines accomplish the same task:

```
put file (sysprint) list ('HI');
put list ('HI');
```

For those of you unfamiliar with logical and physical devices, this may be a little confusing. Let's take a brief look at these terms. A logical device is one that is assigned by the program or a command line, hoping that the device is both there and properly assigned to a physical device. A physical device is one that is a physical part of the user's system such as LPT:, the line printer. These devices are permanently assigned by virtue of being physically cabled for the I/O port to the device itself under logical devices which can be reassigned by the operating system. If you are still confused, rather than getting bogged down in the quagmire of physical and logical devices in this article, refer to any good book on CP/M. The authors Thom Hogan and David E. Cortesi have written two of the best CP/M books available.

Consider the following program fragment showing device file assignment and the use of a file variable:

```
dcl
  output file variable,
  (sysprint,print) file;

open file (print) print title ('$lst');
open file (sysprint) stream print pagesize (0) title
('$con);
.
.
.
.
put skip edit(
  'output to',
  '',
  '1 printer',
  '',
  '2 console',
  '',
  'input choice')
    (7 (a));      /* edit 7 alpha lines */
get list (choice);
if choice = 1 then
  output = print;
    else
      output = sysprint;
.
.
.
put file (output) list (. . . . . );
```

A file variable and an odd edit statement are new concepts. The file variable allows the output to be directed to either the screen or the printer. Note 'sysprint' had to be declared explicitly to assign it to the file variable. This segment shows clearly that all output is file. Let me explain

what is happening in the edit statement. What has happened is that there are seven separate lines of string constants. The (7(a)) tells the 'put edit' statement to print seven strings. The 7 is a repeat factor, one of PL/I's niceties. It is like trapping the parameter following it in a "do for 1 = 1 to 7" statement. Edited I/O is one of those fastidious things that PL/I does like FORTRAN, managing its data into neat columns or rows or whatever is needed by the I/O statements. Edited I/O is dealt with in more detail in **Chapter 6, Edited I/O Plain and Fancy.**

CHAPTER TWO
FUNCTIONS

A function is a form of a procedure that returns a single value. PL/I has a large library of functions that you can tap anytime you wish simply by invoking them programmatically. Incidentally, the full set of PL/I has a monumental function library. It is so large, it would take a dedicated computer writer 20 years to catalog them with any degree of completeness. Seriously, the huge size of PL/I's function library is probably one of the discouraging factors in learning the full set. The function library of Subset G is much more manageable, and Digital Research's version (a subset of Subset G) is about on par with the function library of a good C compiler, with all the functions you reasonably need.

Let's define some terms:

Arguments or Actual Parameters
Variables or constants that are passed to a procedure or function by the calling statement.
call sum(a, b);

Parameters or Formal Parameters
The values received by the procedure or function.
sum: proc (a, b);

Let's look at the following program fragment to examine this concept more closely:

```
hypotenuse = pythagoras(side_1, side_2);
.
.
pythagoras:
  proc (a, b,) returns (float);
    dcl
      (a, b, c) float;
  c = sqrt (a**2 + b**2);
  return (c);
  end pythagoras;
```

The call to 'pythagoras' on the first line passes the arguments, 'side_1' and 'side_2', to the function 'pythagoras' which receives the parameters 'a' and 'b'. 'Sqrt' is also a function, but because it is a part of PL/I's library, it is called a built-in function. The passing of arguments and receiving of parameters is the heart of maintaining local variables and reaching sections of the program that are external to the main body of the code.

PL/I has built-in functions for arithmetic processing; math functions for trig, hyperbolics, logs, and exponentiation; string functions; conversion functions; condition functions for error handling; and a handful of miscellaneous functions. The built-in functions have only to be invoked and passed an argument to return a value. The program statememt 'c = pythagoras (a,b)' passes the arguments 'a' and 'b' to the function 'sqrt' and the value returned is stored in the variable 'c'. Note that arguments can be constants, variables, expressions or even the value returned from another function:

```
c = pythagoras (3,4)
c = pythagoras (a,b)
c = pythagoras (a-5, b+3)
c = sqrt (abs(a))
```

Built-in functions

Here's a look at most of the built-in functions you have at your command in PL/I-80.

Arithmetic Functions

abs
Returns the absolute value of a constant, variable, or expression passed to it.
a = abs(-128) {128}

ceil
Returns the smallest integer greater to or equal to the argument.
a = ceil (9/5) {2}

divide
Divides its first argument by its second to the precision of the third and the scale of the fourth. (Precision is the total number of significant digits, scale (factor) is the number of decimal digits).
Example:
a = divide (62.4,12**3,4,3)
Returns: 0.036

floor
Returns the largest integer not exceeding the argument.
Example:
a = floor (9/5)
Returns: 1

max
Returns the larger of two arguments.
Example:
a = max (4, -128)
Returns: 4

min
Returns the smaller of two arguments.
Example:
a = min (4, -128)
Returns: -128

round
Rounds off its first argument to the right (+) or left (-) of the decimal by the number of places specified by the second argument.
Example:
a = round (1/3,3)
Returns: .333

sign
Returns 1 if the argument is positive, -1 if negative and 0 if 0.

trunc

Returns the integer portion of an argument (like BASIC's INT( ) function).

Example:

```
a = trunc (9/5)
```

Returns: 1

Mathematical Functions

The mathematical functions, most of which are transcendental, are pretty well self-explanatory.

acos	arc cosine, result in radians
asin	arc sine in radians
atan	arc tangent in radians
atand	arc tangent in degrees
cos	cosine from radians
cosd	cosine from degrees
cosh	hyperbolic cosine , radians
exp	returns a value of e raised to the argument
log	returns the natural log of the argument
log2	returns log base 2 (handy for binary expressions)
log10	returns log base 10
sin	returns the sine, radians
sind	sine, degrees
sinh	sine, hyperbolic, radians
sqrt	returns the square root of the argument
tan	returns the tangent of the argument given in radians
tand	tangent, degrees
tanh	hyperbolic tangent, radians

String Functions

When a language is asked to perform string manipulation, it either falls apart or shows its power. FORTRAN, HPBASIC and others die when it comes to strings. PL/I, on the other hand, has a few powerful functions that will do about anything in the way of string handling.

Concatenation is not a function, but I'm throwing it in anyway:

```
new_word = 'horse' || 'feathers'
bad_word = 'horse' !! 'expletive'
```

The pairs "||" or "!!" are concatenate operators.

Here are the string functions:

collate

This is sort of a unique function. It has no parameters but returns the entire ASCII character set in ascending order. You may never use this one, but it sure sounds impressive!

index

This is one of the more powerful and commonly used string functions. It has two arguments, both strings. It returns an integer value showing the position of the leftmost occurrence of the second string in the first.

Example: full_name = 'J S Bach';
position = index (full_name,'Bach')
Returns: 5

(The "sister" function to 'index' is the 'verify' function, which is at the end of the list of string functions.)

length

This function returns the length of the string used as an argument.

substr

The 'substr' (substring) function is perhaps the most heavily worked of all the string functions. It has a string as the first argument followed by one or two integer values. It returns a new string made from the old, beginning at the position specified by the first integer, and (optionally) to the length of the second.

Example:

```
new_word = substr ('horsefeathers',6,7)
```

Returns: feather

translate

This is about as easy to explain as a late payment. The 'translate' function takes three arguments, all string. The third argument is optional and if not used, is assumed to be the entire ASCII collate sequence. The 'translate' function goes through the first string a character at a time, replacing each character with a character from the second argument corresponding to the character in the third argument. You think I wrote that just to screw you up!

Example:

```
translate ('cat', gdo', 'tca')
```

Returns: dog

You know there are a million good uses for this function just waiting to be discovered. . . .

verify

This function takes two strings as its arguments and returns the position of the first leftmost non-occurrence of the second string.

Example:

```
verify ('Sam Slade' , ' ')
```

Returns: 1

You can also create your own functions and add them to the PL/I function library via the PL/I Librarian. Here is a handy function I created to strip leading blanks of numbers after they have been converted to string. It is a very practical function, and you might want to add it to your library:

```
convert:
  proc (number) returns (char(15));
  dcl
    (number,position) fixed,
      (string, string_out) char (15) var;

  string = chr (number);
  position = verify (string, ' ');
  string_out = substr ( string, position);
  return (string_out);
end convert;
```

The 'convert' functions receives the parameter number, an integer, declared fixed. It is converted to character by the function 'chr()' which converts it to

string but leaves a few blanks in the front for the sign and so forth. The problem is to strip the leading blanks. The 'verify' function gets the position of the first non-occurrence of a blank. Again it tells where the first character is that is not the same as the character(s) in the second string. The 'substr' function returns a new string 'string_out' starting at the "position" of the first non-blank, and continuing until the end of the string. The 'convert' function then returns a blank-free alpha representation of the number passed to it as an argument.

Could the 'convert' function have been written with the 'index' function? Not easily. It would have to specify what "character" to look for, 1 thru 0.

Conversion Functions

Data conversion could and should be a chapter in and of itself. PL/I has an entire set of rules on data conversion that have to be understood to do any but trivial conversions. In any conversion from one numeric type to another (or a conversion from numeric to character), the number is first converted internally to character. In the conversion it is padded to the left with a minus sign (–) if it is negative, and it is also padded with three blanks. For example:

128 converts to " 128"
–128 converts to " –128"

Remember our programming tool, the 'convert' function? It was created to get around this padding feature of the language.

When converting from one numeric type to another, PL/I, having first converted to character, now completes the conversion by its own rules. I will try to cover them painlessly. First we will cover numeric conversion functions:

binary
This takes the form
binary (string, precision)
or the form
binary (expression, precision)
If the expression is type float binary, the result returned is also type float. Anything else will have binary return fixed. If the string or expression is decimal then precision is mandatory. Otherwise it is optional.

decimal
This takes the form
decimal (string, int, int)
or
decimal (expression, int, int)
as in
decimal (amount, 9, 2)
The two integers are scale and precision. If no precision is given, then there will be none (no 'decimal places'). Scale can be anything from 1 to 15 binary places. The result of the conversion will be fixed decimal (or binary coded decimal).

fixed
This one is a trick. If it has a character or a fixed decimal argument it returns fixed decimal. Anything else returns type fixed binary. Its form is
fixed (string, scale, precision)
or fixed (expression, scale, precision)
Precision (number of decimal digits) is optional and will be 0 if left out.

float
Straightforward, the 'float' function returns a type float binary number from an argument that may be either string or an expression. The precision is optional.
Example:
float (string, precision)
or
float (expression, precision)

The remaining conversion functions are:

ascii
Remember the 'collate' function? It gets used here. Any number, 1 to 128, passed to the 'ascii' function is returned as its ascii character equivalent.
Example:
character = ascii(number);
The number is found in the collate sequence by its position. The MBASIC equivalent of 'ascii' is 'CHR$()'.

bit
This takes the form
bit (string,length)
or
bit (expression, length)
It converts the string or arithmetic expression to a bit string. 'Length' is optional and if used the bit string will have its length.

character or chr
The 'character' function takes a string or an expression and converts it to character. It has an optional length
chr (exp, length)
which gives a character of specified length, almost. NOTE: Whenever length is used, the length returned will in fact be determined by the compiler as explained in the language manual under conversion rules in the chapter under "Assignments and Expressions."

rank
The 'rank' function is the flip side of the 'ascii' function. It takes as an argument a character and returns its ascii number. Its BASIC equivalent is ASC( ).

unspec
This is an important function. The 'unspec' function takes as an argument an expression of word length (16 bits or less) and returns the contents of the memory address of the expression as a bit string. Unspec can be used as a "pseudo" variable. By that I mean that it can be used on both sides of an assignment (only 'substr' and 'unspec' have this privilege). Let's use the example of a function to return uppercase alphabet characters to illustrate this:

```
upper_case:
  proc (character_in);
    dcl
      (character_in, character_out) char (1);
  if character_in >= 'a' & character_in <= 'z' then
    unspec (character_out) =
      unspec (character_in) & '11011111'b;
    else
```

►

```
    character_out = character_in;
  return (character_out);
```

Short, sweet and heavy! The character passed to 'upper_case' as a parameter is checked to see if it is an 'a', a 'z' or anything in between. If it is, then PL/I demonstrates its ability to go "down and dirty" to system level, to the address of the input character performing a logical and on the contents of the address with 1101 1111. All uppercase characters have the third bit of the first (most significant) nibble set to 0.

Anding it with 1101 1111 will take a lowercase character which will have the third bit set to 1 and set it to 0.

```
0110 0001   'a'
1101 1111   logical and
0100 0001   'A'
```

PL/I has the ability to function at system level, although perhaps not as compactly as C or as its sister language PL/M. Nevertheless, PL/I does some pretty amazing things at systems level, as the 'unspec' function demonstrates. Pointers are one of the most powerful tools in programming at systems level or any other level, and they will be covered later in detail.

Condition Functions

To keep programs from falling on their faces, it is necessary to get into an area of programming called exception processing. PL/I runs a program until an interrupt is encountered. When encountered, if an "enable" is not provided, the program will belly up and die. The condition functions are the "enables."

Way back in files we quickly covered the endfile condition. Let's look at it again:

```
on endfile (prog.dat)
  goto store_file_length;
do i = 1 to 32767;
  read file (prog.dat) into (dummy) keyto (i);
  end; /*do */
store_file_length:
file_length = i;
```

This tool, the above block of code, reads the file and stores its length. Without the endfile function, the program would have given an 'end of file error" and terminated. The 'endfile' function smoothly moves the program execution to the next executable line.

Similarly the 'oncode', 'onfile', and 'onkey' functions aid in keeping program control. They work in conjunction with the "signal" statement.

oncode

The 'oncode' function is a function which returns the most recent runtime error.

Example:

```
on error
  put skip list ('error no ', oncode( ));
```

Returns: a fixed (int) number.

onfile

The 'onfile' function returns the file name for which the most recent endfile or endpage was signaled.

onkey

The 'onkey' function is used with keyed records (key files). This function returns the key (as character) that caused the program to go into an error condition.

These on "conditions" are probably as clear as the L.A. smog to you right now. Hopefully they will become more lucid in the chapter on error processing.

Miscellaneous Functions

addr

This low level function returns the address of the variable passed to it as a parameter. It is invaluable in pointer dependent routines. For example, to get the pointer to a data buffer, you do this:

```
data_pointer = addr(data_structure);
```

Note: The following functions, 'dimension', 'hbound' and 'lbound', deal with the bounds of arrays. When an array is declared, the bounds (limits) are specifically defined. If a lower bound is not specified, it is assumed to be 1.

dimension

This function takes the following form:

```
dimension (array_variable, integer);
```

It returns an integer giving the extent or range of the variable, like this:

```
array_variable(integer)
```

or more simply put, it gives the number of elements.

hbound

The form

```
hbound (array_variable, integer)
```

receives

```
array_variable (integer)
```

and it returns the upper bound of the dimension integer.

lbound

Used like 'hbound', the 'lbound' function returns the lower bound of the dimension integer.

null

The 'null' function is used with pointers and based storage. It gives a zero or null value to a pointer. In practice it is used to show that the list (or other form of data structure) has reached its end. The null signals termination of the list.

The above functions are those resident in the PL/I version marketed by Digital Research. Their version is a subset of PL/I Subset G. Subset G has a substantially larger function library, and the full set has a library that is exponentially larger. For lack of a better word let us say the full set library is elephantine. This enormous function library may well be a detriment to PL/I as a popular language because, traditionally, students resist learning a library (and therefore a language) as large as this. The smaller PL/I-80 library is a convenient size in that it is adequate to perform just about all normal programming tasks without being so large as to discourage learning.

The next installment of this series on PL/I will get into program structure, writing style and all those programming necessities, like branching and decision-making, that make programming fun and functional. ■

Letter to the Editor

I have just finished typing and assembling ERAQ as published in *Lifelines* last September. After I got all of my self-inflicted errors out, it ran like a charm.

The enclosed rearrangement of your code eliminates the trailing comma after the R/O symbol of a file that is "read only" but not "system" (see FILEA.EXT below). Note the change in destination of the JZ instruction on the fifth line of code in addition to the rearrangement of the other lines.

Also, ASM doesn't like " MVI E, -1 " in your "CNVRT:" instruction. It assembles okay but gives a "V" (value) error message. MAC works okay; no error messages. I didn't try RMAC or M80. They're at work and I'm on vacation. With ASM I just substituted " MVI E, 0FFH " which got rid of the error message.

This is from your original ERAQ.ASM:

```
DISP7:  MVI     E, '('
        CALL    OUTPUT
        LDA     ROFLG           ; print R/O symbol?
        ORA     A
        JZ      DISP4           ; nope
        LXI     D, RO
        CALL    STROUT          ; print read/only symbol
                                ; after file name
DISP4:  LDA     ROFLG           ; if printed R/O, then
                                ; print comma
        ORA     A
        JZ      DISP8
        MVI     E, ' , '
        CALL    OUTPUT
DISP8:  LDA     SYSFLG
        ORA     A
        JZ      DISP5           ; print system symbol?
        LXI     D, SYS
        CALL    STROUT          ; yes, print it after file
                                ; name
DISP5:  MVI     E, ')'          ; close symbol string
        CALL    OUTPUT
```

It gives this output:

```
A>ERAQ FILE?.*
FILEA       .EXT    (R/O,)          Erase this file? N
FILEB       .EXT    (R/O,SYS)       Erase this file? N
FILEC       .EXT    (SYS)           Erase this file? N
FILED       .EXT                    Erase this file? N
0 files erased.
```

This is how I modified ERAQ.ASM:

```
DISP7:  MVI     E, '('          ; open symbol string
        CALL    OUTPUT

;       determine if R/O symbol should be printed

        LDA     ROFLG           ; check R/O status
        ORA     A               ; set flags
        JZ      DISP8           ; file is not read only
        LXI     D, RO           ; file is read only
        CALL    STROUT          ; print R/O symbol after
                                ; file name

;       determine if SYS symbol should be printed

DISP8:  LDA     SYSFLG          ; check SYS status
        ORA     A               ; set flags
        JZ      DISP5           ; file is not a system file
        LDA     ROFLG           ; see if R/O symbol was
                                ; printed
        ORA     A               ; set flags
        JZ      DISP4           ; R/O wasn't printed, so
                                ; a comma isn't needed
        MVI     E, ' , '        ; R/O was printed, so
                                ; print a comma
        CALL    OUTPUT
DISP4:  LXI     D, SYS          ; print SYS symbol
        CALL    STROUT

DISP5:  MVI     E, ')'          ; close symbol string
        CALL    OUTPUT
```

It gives this output:

```
A>ERAQ FILE?.*
FILEA       .EXT    (R/O)           Erase this file? N
FILEB       .EXT    (R/O,SYS)       Erase this file? N
FILEC       .EXT    (SYS)           Erase this file? N
FILED       .EXT                    Erase this file? N
0 files erased.
```

It's not that much of an improvement, but it isn't that hard to do either!

Please keep the articles coming. I think I already know more about CP/M than I care to (if I want to keep my sanity), but it's by figuring out how programs like yours work that I learn more when I want to. ■

Don Weisman
Anchorage, Alaska

Tips and Techniques

by John S. Coggeshall

CB-80's CALL statement is completely different from CBASIC's: its argument must be a user-defined function, not an address. For programmers who have invested a lot of code in the CBASIC convention but wish to convert to CB-80, here is a way to circumvent CB-80's requirement and allow transfer of control to an arbitrary address.

Add the following to the main program:

```
        DEF     XFER$( ADDR% )
                EXTERNAL
        FEND
```

and replace 'CALL Address%' with 'CALL XFER$(Address%)'.

Assemble the following into a module compatible with LK80, to be linked to the main program:

```
        PUBLIC  XFER$
XFER$:
        POP     HL              ; Get return-addr, adjust stack
        EX      (SP),HL         ; Swap it with transfer-addr
                                ; (ADDR%)
        JP      (HL)            ; Pass control to ADDR%,
                                ; leaving return-addr on stack
        END
```

Consider this a temporary patch. It may be more efficient to reassemble the old code and link it to the main CB-80 program. It may even be appropriate to rewrite the assembly routines to take advantage of the new standard by getting their arguments off the stack. But now you have a choice. ■

Product Status Reports

New Products

INFORMAX

Abacus Data
1920 San Marco Boulevard
Jacksonville, FL 32207

This electronic file manager offers high speeds, simple screen-driven operation for data storage, retrieval, calculations and reporting. It is user friendly. Up to 14 different data types and formats can be used including: alphabetic, numeric money, phone, date and time, etc. Even phonic searches are supported, allowing retrievals even with incorrect spelling. There is no limit on the number of files or on the number of math operations performed within a record. Each file can contain up to 65,000 records and each record can have up to 50 items or fields. Expansion from one INFORMAX capability to another merely requires purchasing the new product for the price differntial. All data and applications are transportable to higher levels.

Requirements: 8- or 16-bit systems with Z80, 8085, 8086, CP/M-80, CP/M-86, TurboDOS, Mmmost and dpc/OS (MS-DOS, PC-DOS,N-Star and MPM version to be released soon.)

Price: INFORMAX 5 — $395, INFORMAX 10 — $595, INFORMAX 15 — $795, INFORMAX 20 $995

PCT-100

Method Systems Inc.
19751 S. Lakeshore Blvd.
Euclid, OH 44119

The PCT-100 is a configurable in-line RS-232 protocol and data translator, which allows existing software/hardware interfaces to be made compatible and provides system-level enchancements. Two full-duplex RS-232 ports are provided, each configurable from 50 to 19.2K BAUD. Ports are jumper selectable for data set or data terminal operation. Data and protocol information is translated in real time by an internal processor that asynchronously executes a translation algorithm for each port. Translation algorithms are stored in non-volatile EEPROM, which can be write-protected using an external "learn/run" switch. PCT-100 algorithms and port configurations are entered using a built-in editor. The algorithms are coded in MSI's Commmunications Translation Language (CTL). Programs can be easily entered and edited from any ASCII device, including downloading from a host computer.

Price: PCT-100-ASM (Stand Alone Device)—$339
PCT-100-PCB (Assembled & Tested PC Board)—$289

STUDENT RECORD SYSTEM

Single Source Solution
2699 Clayton Rd.
Concord, CA 94519

This file management system aids class management of students and their grades. The STUDENT RECORD SYSTEM has a multiple of menu-selectable functions to aid the instructor and can keep track of 10 separate grades for 100 students, and automatically computes the mean and standard deviation for each graded assignment. Grades on up to 10 assignments may be sorted.

Requirements: CP/M, 8080, Z80, or 8085 or UCSD Pascal 32K min. 43K rec.

Price: $89.95

PROTEX

Single Source Solution
(for address see above)

This text formatter provides a variety of utilities related to outputting text, including repetitive letters, mailing lists, merging external files, special print effects as will as the usual features such as headers and footers, justification and hyphenation.

Requirements: CP/M

Price: $99.50

THE SENSIBLE SOLUTION BOOKKEEPER

O'Hanlon Computer Systems
11058 Main Street 225
Bellevue, WA 98004

This program includes simple, yet complete accounts receivable, accounts payable, and general ledger capability. The more sophisticated The Sensible Solution Bookkeeper Plus adds payroll applications to the package. For example, invoices, purchase orders and other reports can contain up to 384 fields from as many as 10 different files. The Sensible Solution Management Package includes an inventory control system that monitors cost, turn-over, profit and sales, along with sales force tracking. It also includes sales order entry programs for entering and changing orders, creating inventory "pick lists," and developing and printing of invoices. A similar purchase order entry capability is also provided to print purchase orders, record receipts into inventory and check invoices. Other capabilities of the Sensible Solution accounting packages include file and record locking for multi-user operating systems.

Requirements: CP/M, MP/M, MSDOS, DPC/OS, TurboDOS, MmmOST.

Price: Sensible Solution DBMS and Programming Language — $695, Bookkeeper — $495, Bookkeeper Plus — $595, and Management — $895.

MULTED

CompuDraw
1227 Goler House
Rochester, NY 14620

This program searches, edits, prints and manipulates text in multiple files without requiring the user to individually operate each file. It may be used to change or search for the same text entry consistently in a large number of document files, or to change the name of a variable in many program modules in different

files. It is user friendly. Search modes include options such as upper/lower case equivalence, wild card characters, and searches restricted by column position. It can also search for blank lines, or lines not containing a specified text. Text modification options include global changes, the facility to swap words or strings, as well as the option to search and modify specific column positions. Other features include over-write and insert modes. MULTED may also be used to produce listings of one or more files with a single command, complete with file name headings, pagination, manual or automatic paper feed control, optional line numbering and dating. Such listings may be produced on the console, printer or into a disk file. Further, MULTED can optionally produce a log of activity, and direct it to the console, printer or a disk file. MULTED works by accepting English-like commands from the console, or from a disk-file. The latter allows production of command files for regularly performed operations. It then executes these commands on the files whose names are specified by the user at the console or from a disk-file, possibly using wild card characters in each specification. When any input is required from the console, MULTED issues appropriate user-friendly prompts; similarly any abnormal conditions that arise are reported on the console using clear explanatory messages. Other features supported by MULTED include automatic maintenance of .BAK files, alternating between drives for output files, tab expansion or compression, and upper/lower case translation.

Requirements: CP/M, 8080, Z80, 32K, CDOS
Price: $59

VoiceDrive

SuperSoft
P.O. Box 1628
Champaign, IL 61820

ScratchPad with VoiceDrive allows the user to operate the program by speech. This includes all commands and full data entry. The ability to give vocal commands does not hinder the ability to use typed commands. Examples of use include: presentation which require "hands off" control, Non-typist operation, hands-off data entry and retrieval, and handicapped use and training. VoiceDrive is a complete software interface that stands between the voice recognition hardware and application software. It performs two major functions: 1) It performs the basic console and vocal Input/Output, and 2) It translates the spoken instructions into binary codes.

Requirements: PC-DOS version — 128K

Price: ScratchPad with VoiceDrive — $495.

LOAN ANALYZER

Simple Soft
480 Eagle Drive, Suite 101
Elk Grove, IL 60007

This program is an effective tool for analyzing a mortgage or loan. Several professionally formatted reports present concise and tailored analysis. Calculations are included to show complete amortization schedules, effective interest rates, interest paid between dates, the impact of loan charges and the effects of an early loan termination. The program calculates unknown variables such as loan amount, loan term, loan payment and ballon payments. LOAN ANALYZER is the newest addition to the QUIKCALC line of financial tools.

Requirements: VisiCalc or SuperCalc

Price: $99.95

New Books

Crash Course in Digital Technology

Howard W. Sams & Co., Inc.
4300 West 62 Street
PO Box 7092
Indianapolis, IN 46206

Crash Course in Digital Technology, by Louis E. Frenzel, Jr., is written in a programmed learning format using brief informational frames and frequent self-tests; it is well-adapted to both personal learning and classroom instruction. It is a learning tool, rather than a reference, and is well-illustrated with photographs, diagrams, and examples. With Frenzel's typically easy-to-read style, **Crash Course in Digital Technology** covers what "digital" means, how it represents real-world phenomena, the devices used in handling data, how these devices perform logical operations, how these are combined, and more. The instruments used to troubleshoot digital circuits are also discussed.

Computer Selection Guide

Para Publishing
PO BOX 4232-2
Santa Barbara, CA 93103-0232

Computer Selection Guide, Choosing the Right Hardware and Software: Business-Professional-Personal, by Dan Poynter, proves to be a short course in computers so you will understand what you are getting into, checklists so you are sure to consider every important function and feature, and a source book so you can find what you want. A special bonus chapter shows how to set type with a word processor or computer.

Word Processing and Information Processing

Para Publishing
(see above address)

Word Processing and Information Processing, by Dan Poynter, tells what these new machines are, describes how they work, shows how you can benefit from them, translates the new "computerese," and provides numerous buying tips.

Before You Buy A Computer

Crown Publishers, Inc.
One Park Avenue
New York, NY 10016

Before You Buy A Computer, by Dona Meilach, is a handholding guide based on the author's research of the marketplace. It is a guide to buying your first computer and is designed for the layman who doesn't know what questions to ask and might be intimidated by a first meeting with a computer salesperson. Whether you shop for your computer tomorrow or three years from now — for you personal or office use — **Before You Buy A Computer** will provide you with research methods that will never be out-of-date. Along with 30 black-and-white

photos and drawings, there are chapters with assignments requiring you to keep popping in and out of stores until you're ready to pass a computer literacy course with flying colors. Promotional material prepared by Crown's publicity department is available.

New Versions

SuperSoft C COMPILER FOR CP/M-80

SuperSoft
1713 S. Neil Street
P.O. 1628
Champaign, IL 61820

The new release is syntactically compatible with UNIX version 7 C, and supports such features as long integer functions and double floating point functions (including trigonometric functions). An extensive list of library functions is provided with source code.

SuperSoft C is a multi-pass compiler which produces highly optimized code, making it possible to avoid assembly language coding for most tasks. The compiler is fast in both compilation and execution. Many UNIX compatible functions have been included allowing the porting of source between UNIX C and SuperSoft C with few, if any, source changes.

Price: $275, for CP/M-80 version; $500 for other versions.

MAIL 80™

Pegasus-Basis Software, Inc.
670 International Parkway
Suite 100
Richardson, TX 75081

This new release features an installation program which configures the system for a variety of microcomputers and video terminals. The library includes terminal characteristics for ADDS, Televideo, Xerox, Soroc, Hazeltine, Zenith and others. In addition, almost any terminal can be installed by answering the prompts which ask for the needed control sequences.

Price: $295.

NEW VERSIONS

PMATE-PC	3.24
MEMORY/SHIFT (for IBM PCDOS)	v2.0
BSTAM-86 (for CP/M-86 DEC Rainbow)	v4.6
FLOAT-87 (for LATTICE "C" and BASIC-86)	v1.0
XLT-86	v1.0
UNIVAIR series 9000 Demo's:*	
DENTAL	2.07
MEDICAL	2.07
INSURANCE	2.07
LEGAL TIME	2.0
LATTICE "C" Compiler for CP/M-86	v1.04/1.25

*will run under CP/M-86 if customer has CBASIC-86

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.

CBUG
Joe Butler
3208 Magicwood Circle
Sacramento, CA 95827
This is for C users. They also publish a bi-monthly newsletter of twenty or thirty pages and include a diskette of software. The cost is $20 a year for six issues.

CBNews
Software Magic
11669 Valerio Street –213
North Hollywood, CA 91605
Their charter is to bind together any and all information of use to the myriad users of Digital Research's compiled CBASIC. They want to spread the word about CB-80, about its bugs and its features and, in general, to make available all new information on CB-80. The newsletter is $12 for 12 issues.

Ferox Products Users Group
1701 North Fort Myer Drive 6FL
Arlington, VA 22209
This group is for experienced and inexperienced users alike. Members learn from each other's successes and frustrations in using Ferox products. The users group will also provide a wealth of information for members on related topics such as compatible software and new hardware devices available. Finally, the users group will provide a forum to discuss product enhancements and other matters that can later be presented to Ferox for follow up. A new group is forming in New York City.

Manhattan Micro
c/o William Wahlin
144 West 86 Street
New York, NY 10024
This group is for anyone interested in IBM-PC. The location presently is the Bramson ORT Technical School at 44 East 23 Street, 8th floor, Classroom 2. Our meetings are usually small, and those present at a meeting will usually determine what they want to talk about, be it hardware, software, assembly language, graphics, or whatever. Occasionally we have a speaker. There are club diskettes of programs which we compile from various sources, sometimes containing the work of our members, other times holding programs we have obtained from bulletin boards. These are a popular product. Dues are $15 per year.

C Users Group
Box 287
Yates Center, KS 66783
The C Users Group is open to all C users. We began as the BDS C Users Group and thus much of our library and discussion centers on that compiler. We hope, though, to expand and grow as the use of C expands and grows. We are seriously interested in the adaptation of our library to other compilers and operating systems and we encourage all persons with a serious interest in C to join and participate. Ownership of BDS C is definitely NOT a prerequisite of membership.

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 84, one of CPMUG's most popular volumes:

NUMBER	SIZE	NAME
.....	.....	.CATALOG.084
CONTENTS OF CP/M VOL.084		
.....	.....	.ABSTRACT.084
Abstracts of files on this disk		
.....	.....	.CRCK.COM
Program for checking CRCs of files		
.....	.....	.CRCKLIST.084
CRCKs of files on this disk		
084.1	16K	MODEM7.DOC
Documentation for MODEM program (from CP/M UG Vol. 79)		
084.2	14K	MODEM76.LIB
Macro library used with MODEM 7.65		
084.3	6K	MODEM76.SET
Instructions for "hot-patching" MODEM		

►

084.4 . . . . . 63K MODEM765.ASM
Macro assembler source code for MODEM 7.65
084.5 . . . . . 10K MODEM765.COM
Object code of MODEM 7.65
084.6 . . . . . 18K SEQIO22.LIB
Macro library used with XMODEM 5.0
084.7 . . . . . 3K XMODEM47.DOC
Documentation for XMODEM program
084.8 . . . . . 49K XMODEM50.ASM
Assembler source code for XMODEM 5.0
084.9 . . . . . 1K XMODEM51.FIX
Notes on a bug fix for XMODEM

Note: MODEM program requires assembly with Digital Research MAC macro-assembler. XMODEM may be assembled with ASM.COM if the "logging" feature is disabled, otherwise MAC is required for assembly.

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

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

Oops!

For those of you who were about to throw your microcomputers out the window because you couldn't get Steven Fisher's listings ("Auto-Start Your CP/M," June 1983, and "Take Control," July 1983) to work, STOP! Several figures were incorrectly printed in the text. Read on for the *correct* listings.

June 1983 issue

Figure 2 — Proper BIOS Implementation For Auto Start

```
MSIZE   EQU   62              ; Size of system in K
BIAS    EQU   (MSIZE-20)*1024 ; Offset to minimum 2.2
CCP     EQU   BIAS + 3400H    ; Base of CCP
BDOS    EQU   CCP + 0806H     ; Base of BDOS
BIOS    EQU   CCP + 1600H     ; Base of BIOS

GOCPM   MVI   A,0C3H          ; JMP instruction
        LXI   H,WBENT         ; BIOS Warm Boot Entry
        STA   0001H
        SHLD  0001H           ; Warm Boot linked
        LXI   H,BDOS          ; BDOS Entry
        STA   0005H
        SHLD  0006H           ; BDOS linked
        LXI   B,0080H         ; Default Buffer
        CALL  SETDMA
```

Figure 4 — Inserting an Auto-Start Command in the CCP

By using the standard utility programs supplied with your CP/M system, you can create your own command to be used when your computer is first turned on. For instance, to have your computer automatically use the command "**A:AUTO**" (06 41 3A 41 55 54 4F 00), just type what is underscored:

```
A>A:SYSGEN
Source Disk? (press RETURN to skip): A
Insert Source Disk in A, then press RETURN when ready:
Destination Disk? (press RETURN to skip): <cr>
```

July 1983 issue

```
A>DDT CONTROL.COM
NEXT PC
0100 0200
-- S0101          select which device is controlled
--  05 05         (02 = console, 04 = aux, 05 = list)
-- 21 .           (a period stops memory substitution)
-- S0114          (enter the command length and text)
-- 00 01
-- 00 OC
-- 00 .           (stop entry with a period)
-- G0000          (reboot, leaving program in memory)
A>SAVE 1 FORMFEED.COM
```

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™
1651 Third Ave., New York, New York 10028

Second Class Postage Paid
At Smithtown, N.Y.

EXPIRATION DATE: 12/83

1