


# PolyLetter



---

## The System-88 Users Newsletter

PolyLetter 92/1

Page 1

JAN/FEB 1992

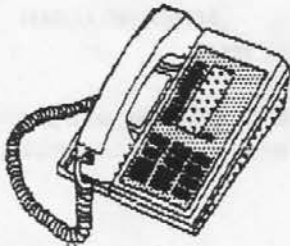
### Editorial

Well, it turns out that the BarCode article is getting to be quite involved. It has a lot of technical details of assembly language programming. We have done less of this in recent issues, so I guess it's about time. What do you think? Getting anything out of something like this once and a while? I'd like to do more survey-interviews, but I need users to send in their survey sheets. How about it people? Send in some more info on yourselves and your use of poly. (The survey form was printed in *PolyLetter* 91/1.)

---

### Communications

Jim Brennon runs an astronomy oriented bulletin board (916) 652-5920. (Voice 916-663-3568.) Jim, who is an old time Poly user, has decided to set aside a Poly area on his bulletin board. I will eventually be uploading some of the Public domain programs to the bulletinboard.



---

### by Leap(Year)s and Bounds

Here's a basic question. When is the truth negative? When an IF test is true in Poly BASIC the value returned is 1. We sometimes use this fact to compute numbers. For example, a leap day is added to years that are divisible by four, except for even centuries, unless those centuries are divisible by 400. Confused? If you are, you aren't alone. In computing a calendar one must make an adjustment for leap days on leap years. Testing to see if a year is divisible by 4 is fairly easy. We divide the year by 4 and see if the result is a whole number.

In BASIC there is a function which rounds a number down to the nearest whole number -- INT. INT(X) returns the *integer* part of the number. INT(2.6) returns the value 2. All we have to do is see IF the number is the same as its integer part. If X=INT(X) is true, then the number X was a whole number. How can we use this fact testing to see if the current year is a leap year? Our first test is to see if it is evenly divisible by 4. If it is, then the result will be a whole number. So, if we use variable Y for the year, our test is

```
IF Y/4=INT(Y/4)
```

But in Poly BASIC we can use our knowledge that "truth is positive". A true condition returns the numerical value 1 and a false condition returns the numerical value 0.

```
>PRINT (1=1)
1
>PRINT (1=0)
0
```

So, if we wanted to write a very simple program to tell how many days are in a given year, we could start with something sneaky like the following.

```
10 INPUT "What year? ",Y
20 PRINT "Year",Y," has",.365 + (INT(Y/4)=Y/4)," days.
```

Knowing that the condition (INT(Y/4)=Y/4) is 1 if Y is evenly divisible by 4 and 0 if it is not allows us to add the leap day automatically.

Unfortunately, every year that is divisible by 4 is not always a leap year. Years that are an even century are not leap years. 1900 was not a leap year. But we can use the same basic technique in a slightly more complicated test. We will test for years that are divisible by 4 and not divisible by 100. We already have the first part of the test. The second part, not divisible by 100 is very similar, except we are

dividing by 100 instead of 4 and our test is not equal rather than equal. To express this condition in BASIC we get  $Y/100 <> INT(Y/100)$ . Now we can express the improved condition as

```
Y/4 = INT(Y/4) AND Y/100 <> INT(Y/100)
```

Because relational operators are evaluated before logical operators, no additional parentheses are needed. BASIC will correctly compute the two conditions and then compute the logical AND for them. If both are true, BASIC will return 1 for the result. Because BASIC gives higher priority to addition we must put the whole condition in a set of parenthesis. Our new and improved program might work like this.

```
10 INPUT "What year? ",Y
20 X=365 + (Y/4 = INT(Y/4) AND Y/100 <> INT(Y/100))
30 PRINT "Year",Y," has",X," days.
```

If we did not include the parentheses, BASIC would add  $Y/4$  to 365 first, and that would give an incorrect result.

But we still aren't done. This computation does not take into consideration when centuries are divisible by 400. Centuries divisible by 400 are leap years. The year 2000 will be a leap year, and the above program doesn't work for the year 2000. We need to add the a test for the condition of centuries divisible by 400. By now you get the idea. The condition in BASIC is " $Y/400=INT(Y/400)$ ". But when we add this condition this time, we must make sure BASIC evaluates the LOGICAL operators in the correct order. We want years divisible by 4 and not divisible by 100, or years divisible by 400. To insure this is evaluated correctly we put the first test in parenthesis.

```
(Y/4=INT(Y/4) AND Y/100<>INT(Y/100)) OR Y/400=INT(Y/400)
```

Our final program:

```
10 INPUT "What year? ",Y
20 L=(Y/4=INT(Y/4) AND Y/100<>INT(Y/100)) OR Y/400=INT(Y/400)
30 PRINT "Year",Y," has",X+L," days.
```

Now, what has all this got to do with "negative truth", you ask? Well, it turns out that on the PC, a condition test returns -1 when it is true. The program above runs perfectly under both Poly BASIC and GWBASIC under DOS, but the result under DOS gives 364 days when the Poly says 366 days! (I know, it runs perfectly, it does not run correctly.) We could

"fix" this by just multiplying the result by -1 on the PC version. But, to my way of thinking, that is not a very aesthetic solution. I'd like to have one program which runs correctly on both systems. Fortunately, there is a way to do that.

The above programming technique is considered by some to be a "trick" which is not a straight forward use of the programming language syntax. It depends upon knowledge of how logical conditions are represented numerically. We should, according to these complainants, not use conditions in numerical computations. We should use conditions only where conditions are specified, in IF statements. The syntax is:

```
IF <condition> THEN <statement>
```

In the current context, we are essentially doing the computational equivalent of the following logic.

If it is a leap year then add 1 to the number of days.

To be true to the formalists, we should write our program so that it basically looks like the above logic.

```
N=365
IF (Y/4=INT(Y/4) AND Y/100<>INT(Y/100)) OR
Y/400=INT(Y/400) THEN N=N+1
```

If we add this coding to the program we get one which runs perfectly and correctly on both systems (almost).

```
10 INPUT "What year? ",Y
20 N=365
30 IF (Y/4=INT(Y/4) AND Y/100<>INT(Y/100)) OR
Y/400=INT(Y/400) THEN N=N+1
40 PRINT "Year",Y," has",N," days.
```

Unfortunately, the PRINT statements work a little differently in both BASICS, so the PC version must move a few spaces. Last issue I showed how to get around this. Here is one way:

```
10 INPUT "What year? ",Y
20 N=365
30 IF (Y/4=INT(Y/4) AND Y/100<>INT(Y/100)) OR
Y/400=INT(Y/400) THEN N=N+1
40 O9="Year"+STR$(Y)+" has"+STR$(N)+" days."
50 PRINT O9
```

Unfortunately, we must add 5 DIM O9(1:64) to the Poly version in order to see the entire output. I'll talk about dimensioning differences in another article. Well, do you think we've leaped to enough conclusions yet? I started this in the Bit Bucket, but it got too big and became an article of its own, yet I still have a bit of the bucket flavor in it.

**Postal Barcodes (part 2)**

In the last issue we determined that, for my particular printer, I needed to print posnet barcodes using 1 column of dots separated by 3 blank columns, and that the tall bars would use 10 dots while the short bars would use 4. The rest of the task will be in the programming effort.

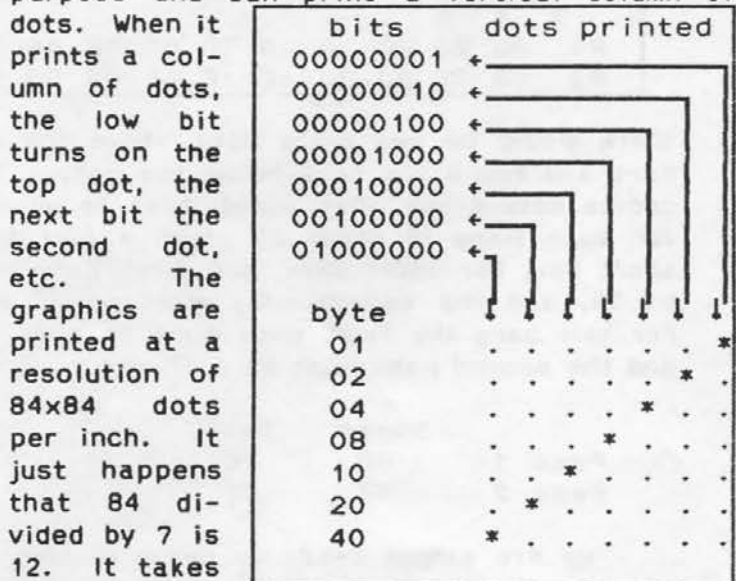
I already had a label printing program which reads a data file and which prints labels. I decided to modify this program by adding the barcode in place of a blank line. This program is an assembly language program, so we will be able to exercise direct control over the printer. Ok, on with the actual programming itself.

Before we get to the details of printing the actual bars, we need to look at how digits are represented by bars. Each digit is represented by 5 bars consisting of 2 tall ones and short ones. Which bars are tall and which ones are short can be figured out using a quasi-

digit	7	4	2	1	0	barcode
"0"	1	1	0	0	0	
"1"	0	0	0	1	1	
"2"	0	0	1	0	1	
"3"	0	0	1	1	0	
"4"	0	1	0	0	1	
"5"	0	1	0	1	0	
"6"	0	1	1	0	0	
"7"	1	0	0	0	1	
"8"	1	0	0	1	0	
"9"	1	0	1	0	0	

binary scheme. We can think of the 5 positions of the bars as encoding numerical values. The right most position encodes a zero, sort of like a parity bit. The next one encodes a 1, the next encodes a 2, and so forth, except that the left-most one, which would encode an 8 in a binary system, actually encodes a 7. The table at the bottom of the left column shows the corresponding relations.

My printer implements graphics using a dot column format. 7 wires are used for this purpose and can print a vertical column of dots.



etc. The graphics are printed at a resolution of 84x84 dots per inch. It just happens that 84 divided by 7 is 12. It takes 12 passes of the print-head to print one inch worth of graphics. Right away we can see that it will take two passes to print the tall barcode bars.

But two passes out of 12 is 1/6th of an inch and is just right for 6 lines per inch. We will have to get the printer to print 1 pass of dots, the top part of the tall bars, advance by only one half a normal line, and then print the bottom part of the tall bars as well as the short bars. To help in preparing for programming printing the codes, I made a chart showing all possible dot positions for the 2 passes.

With this knowledge, I can decide what combination of dots to use to print the short and tall bars. I drew a diagram showing each way I could fit a tall and a short bar in two passes and then picked the one I liked best. Here's what they look like. These are the ways that short and tall bars could be printed using 4 and 10 dots in 14 dots. I marked the ones I chose.

I chose to implement the barcodes by centering the bars in the two passes. That way

P 1	.	.	.	.	.	.	.	.	.	*
A 2	.	.	.	.	.	.	.	.	.	*
S 3	.	.	.	.	.	*	.	*	.	*
S 4	.	.	.	*	.	*	.	*	.	*
- 5	.	*	.	*	.	*	.	*	.	*
1 6	.	*	.	*	.	*	.	*	.	*
7	.	*	.	*	.	*	.	*	*	*
P 1	.	*	.	*	.	*	*	*	*	*
A 2	.	*	.	*	*	*	*	*	*	*
S 3	.	*	*	*	*	*	*	*	*	*
S 4	*	*	*	*	*	*	*	*	.	.
- 5	*	*	*	*	*	*	.	.	.	.
2 6	*	*	*	*	.	.	.	.	.	.
7	*	*	.	.	.	.	.	.	.	.
P1	00	70	00	78	00	7C	00	7E	40	7F
P2	78	7F	3C	3F	1E	1F	0F	0F	07	07

there would be two blank dots above the tall bars and two blank dots below the code. This choice determines what bytes must be printed for each pass in order to print a tall or a short bar. For short bars, the first pass must be 00, and the second pass must be 1E hex. For tall bars the first pass must be a 7C hex and the second pass must be a 1F hex.

	Short	Tall
Pass 1	00	7C
Pass 2	1E	1F

We are almost ready to begin writing the code itself. I think an easy way to do it would be to create a bufer with the graphics data in it and then send the whole batch to the printer. Actually, there is quite a lot of detailed drudgery anytime one write assembly language code. But the experience can be quite rewarding, especially when the bugs have been exterminated and the program actually works. Ok, let's review what we must put in the buffer. First, we must instruct the printer to switch to graphics. Second, we must insert the first tall framing bar. Thirdly, we must convert each digit into its sequence of bars and put those in the buffer. Next, we must insert the checksum digit. And then we must insert the last framing bar. Finally, we must tell the printer to switch out of graphics mode and do a carriage return. But wait! Now we must do a half line-feed and do the same thing over again for the second pass.

I think we can get some structure out of this. Obviously, we can make one routine to do each pass if we have a way to tell it to use different bytes for each pass (the top bytes for the first pass and the bottom bytes for the second pass). We should also be able to use a

routine to build a digit's worth of barcodes.

There are two pointers we are going to have to keep track of. One is a pointer to the characters that make up the zipcode we are processing. The other is a pointer to the buffer where we are building the graphics data. I chose to make HL point at the source digits and DE point at the destination buffer.

Remember, one bar is going to be printed as one column of dots followed by 3 columns of blanks. Since we must do this often, let's make a subroutine to do it. Assume we have the code for the bar in the accumulator; we stuff it in the buffer and then stuff 3 zeros in after it.

```

Stufc  STAX D      ;Put in the bar data
       INX D      ;and move up
       XRA A      ;zero the accumulator
       STAX D     ;Put in one null byte
       INX D     ;bump
       STAX D     ;second null byte
       INX D     ;bump
       STAX D     ;third null byte
       INX D     ;bump for next time.
       RET       ;go back.

```

We can call Stufc when we want to insert the code for a single bar. To do each digit, we will have to get the byte for its bars, process that so we can pick out whether we are going to have a tall bar or a short bar, and then call Stufc for each bar. Before we do that, we are going to have to have a table of the codes for each bar. I just copied the codes listed above into a table using binary notation for easy reference.

```

;                               ;ZipCode digits
Table DB 11000B                ;"0" = 30H
      DB 00011B                ;"1" = 31H
      DB 00101B                ;"2" = 32H
      DB 00110B                ;"3" = 33H
      DB 01001B                ;"4" = 34H
      DB 01010B                ;"5" = 35H
      DB 01100B                ;"6" = 36H
      DB 10001B                ;"7" = 37H
      DB 10010B                ;"8" = 38H
      DB 10100B                ;"9" = 39H

```

Using the table we can look up the bar data for a digit. Since the position in the table corresponds to the zipcode digit, we can use the digit itself to look up the code. As before, let's assume that the zipcode digit value to be converted is in the accumulator. I

have included the ASCII hex value of the digits in the comments for the table. Notice that if we can just get rid of the high nyble of the byte we would be left with the number of the digit. We can do this by subtracting 30H, or just by killing the leading 4 bits by ANDing the actual zipcode digit with 0FH. For example 33H-30H yields 03H; 33H AND 0FH yields 03H. The resulting number is also the offset into the table.

We can devise a translate routine, which I'll call Xlat, which will start with the zipcode digit, convert it to its numerical value, and then lookup and get the corresponding table entry. We will be calling this routine for each digit of the zipcode, so we can also add a routine to add the digits for computing that checksum digit. We will presume that register C will contain the working sum. To point into the table, we are going to have to use one of the index registers DE or HL, but both of them are in use, so we must save and restore the one we choose to use. I chose DE. We point into the table by adding the offset to the address of the table. Because 8080's are 8 bit machines, we must do the low and high bytes separately.

```
Xlat  PUSH D      ;Save DE
      ANI 0FH    ;Convert ASCII to number
      MOV E,A    ;Save it here

      MOV A,C    ;Get the checksum
      ADD E     ;add this digit
      DAA      ;Adjust for decimal
      MOV C,A   ;Save the checksum

      MVI A,Table AND 0FFH ;Low address
      ADD E     ;Add the offset
      MOV E,A  ;Replace it
      MVI A,Table/100H ;Hi address
      ACI 0    ;for role over
      MOV D,A  ;DE now points at
      LDAX D   ;at data so get it
      POP D   ;Restore this
      RET     ;go back
```

The Xlat routine takes a zipcode digit in the accumulator and returns the byte which shows which bars are tall and which are short, in order, from left to right. A tall bar is represented by a 1 and a short bar by a 0.

I will shift the data left then look to see if the leftmost bit is a 1 or a zero. I will do this once for each of the 5 bars. Depending upon whether the leftmost digit is a 1 or a

zero, I will stuff the data for a tall or a short bar into the buffer. Since Zipcode bars only use the lower 5 digits, I will need to do 3 left shifts first. I could use RAL or RLC to rotate the accumulator left, but those routines do not set the sign flag which tells whether the leftmost bit is a zero or a 1. ADD A will do both jobs at once. (Adding a binary number to itself is the same as multiplying it by 2, and that is just shifting all the digits left.) Since this routine is going to process a whole digit, I'll call it StufDig. StufDig can't just call Stufc because we still need to put in either the high or the low bar graphics data. Also, the value of the accumulator, the barcode digit data, must be saved. A PUSH and a POP will take care of saving it, but we need a subroutine to get the proper graphics data. How's Gd (Get data) for a name? Remember, the ADD A set the sign flag, so a positive value is a short bar (0) and a negative value is a tall bar (1).

```
Gd    MVI A,ShortT
      RP      ;Positive so give back short
      MVI A,TallT
      RET    ;Negative so give back tall
```

ShortT and TallT will be set to the graphics byte value for short and tall bars respectively. Putting these together gives us Stufit, which saves the accumulator with a PUSH, call the Gd routine to get the proper graphics byte and then calls the Stufc routine to put that byte into the buffer. Finally a POP restores the accumulator.

```
Stufit PUSH PSW    ;Save barcode data
      CALL Gd     ;get the graphics data
      CALL Stufc  ;insert in buffer
      POP PSW    ;Retrieve barcode data
      RET       ;go back
```

Ok, now we can take a look at the StufDig routine all together.

```
StufDig ADD A     ;00XXXXX0 <- 000XXXXX
      ADD A     ;0XXXXX00 <- 00XXXXX0
      ADD A     ;XXXXX000 <- 0XXXXX00
      CALL Stufit
      ADD A     ;Second bar
      CALL Stufit
      ADD A     ;Third bar
      CALL Stufit
      ADD A     ;Fourth bar
      CALL Stufit
      ADD A     ;Fifth bar
      Stufit  PUSH PSW
      CALL Gd
```

```
CALL Stufc
POP PSW
RET
```

StufDig starts with an actual zipcode digit in the accumulator. It converts that into the byte data for the bars and stuffs the graphics bar data into the buffer, including the nulls between bars.

This code illustrates an interesting technique. The tail of the routine is called as a subroutine. It saves 4 bytes of code code by eliminating a CALL and a RET.

So far, so good. Now we've got to use this routine to process all 5 or 9 zipcode digits and the checksum digit. Let's assume some other routine actually checked and counted the zipcode digits and we have either 5 or 9 stored in ZipSiz. We have already assumed that register C is to be used for the computation of the checkdigit, HL is used to point at the zipcode buffer, and DE is used to point at the graphics data buffer. To process all 5 or 9 zipcode digits we must start with the count of the digits and a zero checksum. Then we get a digit from the zipcode buffer, translate it, and then use StufDig to process it. We must loop while counting down the number of digits. Finally, we must adjust the checksum digit and stuff it in the buffer (with StufDig).

```
DoZc   LDA ZipSiz   ;Get count (5 or 9)
        MOV B,A     ;Initialize counter
        MVI C,0     ;Clear the checksum
; ** HL points at the zipcode buffer **
SI     MOV A,M     ;Get a zipcode digit
        INX H       ;move up to the next
        CALL Xlat   ;Translate it
        CALL StufDig ;Stuff bars in buffer
        DCR B       ;Count down
        JNZ SI      ;More to do so loop
        MOV A,C     ;Get checksum dig
        ANI 0FH     ;Trim to last Dig
        CMA        ;Convert it to
        INR A       ;a negative number
        ADI 10      ;add 10
        CALL Xlat   ;translate the checkdigit
```

Here is where we can just put the routine StufDig, or we can JMP StufDig to finish the checkdigit.

So far we have covered the code need to put the actual postnet barcode graphics data for a 5 or 9 digit zipcode, including the checkdigit, into a graphics data buffer, but we have

included the code for switching into and out of graphics, nor the code for the beginning and ending framing bar. We also need to figure out how to use the above code twice, once for each graphics pass. Let's start with one pass including the graphics data.

My printer uses ETX (03H) to switch it to graphics mode, so the first byte must be a 3.

```
Graph  MVI A,3     ;Turn on graphics
        STAX D
        INX D
```

I also want to have a little bit of space in front of the barcode data. One way to do that is to stuff a blank bar in the front of the buffer. I do that by using Stufc to put 4 zero bytes into the buffer.

```
XRA A           ;Leading space
CALL Stufc
```

Next we must insert the leading frame bar. Stufit puts a tall bar in if the minus flag is set. We can set the flag using the logical OR on a byte with the sign bit set to 1.

```
ORI 80H        ;Leading frame bar
CALL Stufit
```

Ok, now that we have the leading frame bar in the buffer, we can put the zipcode bars in by calling DoZc

```
CALL DoZc      ;Insert zip code data
```

Our trailing frame bar can be done just like the leading frame bar.

```
ORI 80H        ;Trailing frame bar
CALL Stufit
```

The printer must be switched out of graphics mode with an ETX (03H) and a STX (02H).

```
MVI A,3       ;End...↓
STAX D
INX D
MVI A,2       ;graphics
STAX D
INX D
```

After switching out of graphics mode we must send the printer a code to do a half line-feed and a carriage return. For My printer that is just the byte 0EH.

```

MVI A,0EH      ;1/2 line feed
STAX D
INX D
RET

```

That gives us one pass. We need to look at the routine as we now have it to see how it can be used to print both passes. But, let's save the rest for later. (To be continued.)

### Computing your Mortgage

I have both printer mini-cards installed. One is connected to a printer with paper for writing letters, etc. The other is connected to a printer with labels. With this configuration, I can write programs which automatically switch between printers as needed. Such a program is the one I use to compute my mortgage payment each month. I usually pay more than the minimum amount due, so I can't depend upon the coupon book for accurate data. So I wrote a program to not only compute the amount of interest and principle paid, but to keep track of it in the program itself. The program prints out two copies of a mortgage account statement, switches to the label printer, and then prints out a label for mailing the payment.

This program also treats itself like a data file and updates the data lines in the program itself. I'd like to see a PC BASIC program do that! (I'll have to look into that for a future article.)

The first 7 lines of the program serve as the data file for the program. Any time I want to see what the current status is, I just need to type the file.

```

10 DATA 40000.000 \REM Balance      ;
20 DATA 8.500 \REM Interest rate;
30 DATA 347.120 \REM Amount due    ;
40 DATA 400.000 \REM Amount paid   ;
50 DATA 000.920 \REM Interest YTD ;
60 DATA 000.080 \REM Principle YTD;
70 DATA 1992.030 \REM Date        ;
80 REM Above data maintained by the program.

```

Since this is a financial program I want plenty of accuracy, so I set the number of digits to 20. P\$ is for sending data to the printer, and I will talk more about that later. The first thing the program does is to read in the data. Since the data is already in the program itself, it does not need to open a file (yet).

```

90 DIGITS 20
100 DIM P$(1:63)
110 READ B0 \REM Balance
120 READ I0 \REM Interest rate
130 READ P0 \REM payment due
140 READ T0 \REM total paid
150 READ Y1 \REM Interest YTD
160 READ Y2 \REM Principle YTD
170 READ D0 \REM Date

```

We're going to need month names, so some DIM statements, MAT READ statements, and DATA statements are required.

```

180 DIM M0$(12:3),M1$(12:9) \MAT READ M0$,M1$
190 DATA "JAN", "January", "FEB", "February", "MAR", "March"
200 DATA "APR", "April", "MAY", "May", "JUN", "June"
210 DATA "JUL", "July", "AUG", "August", "SEP", "September"
220 DATA "OCT", "October", "NOV", "November", "DEC", "December"

```

Once this preliminary stuff is taken care of, the program reminds me that I must have the printers connected, and then goes about setting up the printers. Line 260 connects the letter printer; lines 270 to 320 sends the codes to setup the printer for some printing format conditions. The codes are actually sent to the printer one character at a time through Worm Hole 5 (WH5=0C34H). This bypasses the part of the printer driver which counts characters.

```

230 PRINT "This program prints a Mortgage statement."
240 PRINT " The printers must be connected for it to work."
250 PRINT "I'm connecting the letter printer.",
260 P$="tiger"+CHR$(13) \Z=CALL("Prnt",2,0,0,MEM(P$))
270 P$=CHR$(29)+CHR$(6)+CHR$(5)
280 P$=P$+CHR$(27)+"R,1019" \REM Letter quality
290 P$=P$+CHR$(27)+"B,89" \REM 6 lines per inch
300 P$=P$+CHR$(27)+"J,120,9009" \REM Set 1 inch margins
320 FOR I=1 TO LEN(P$) \X=ASC(P$,I) \Z=CALL(3124,X) \NEXT

```

Line 330 sets up the lines per page, characters per line, and top and bottom margins.

```
330 Z=CALL("Prnt",7,66*256+65,6*256+6,0)
```

Remember that I said the program treats itself as a data file? Well, we do have to open the program as an inout file in order to change the stored data statements. Line 340 does that.

```
340 FILE:2,LIST \FILE:4,OPEN,"BS<MORT.BS",INOUT
```

The next thing we do is to compute the year (Y0) and month (M0) from the stored date

code (D). Since the stored date is the date of the last payment, we must increment the month and see if we pass December.

```
350 Y0=INT(D0) \REM Year
360 M0=(D0-Y0)*100 \REM Month
370 M0=M0+1 \IF M0=13 THEN GOSUB 550
```

The program keeps track of the balance on the mortgage, the interest rate, the monthly payment due, the interest and principle paid during the current year and the date. The next task is to update these amounts for the current payment. Since the interest is stored in the familiar (to us) annual percent rate, it must be converted to a form suitable for computing the monthly amount due. We divide it by 12 months. We must also round the result to the nearest whole cent. Since interest, as a percent, is already multiplied by 100, we can just take the integer part of the result. Adding .5 takes care of rounding. But the result is in cents, so we must divide by 100. Line 390 takes care of this.

```
380 B1=B0 \REM Opening Balance
390 I1=INT(B1*10/12+.5)/100 \REM Interest due
```

Once we have computed the interest due, we can compute the amount which will go towards the principle by subtracting the interest from the amount of the monthly payment. Line 400 does this.

```
400 P1=P0-I1 \REM Principle due
```

We can compute the additional principle by subtracting the payment due from the actual total we decide to pay (Line 410). To get the total principle paid this month, we must add this to the principle due (Line 420).

```
410 P2=T0-P0 \REM Additional principle
420 P3=P1+P2 \REM Principle paid
```

Of course, the new balance will be the old balance less the principle paid.

```
430 B2=B1-P3 \REM New balance
```

And the cumulative (Year To Date) interest will be the previous interest plus the current interest. Similarly with the principle.

```
440 Y1=Y1+I1 \REM Interest YTD
450 Y2=Y2+P3 \REM Principle YTD
```

Once these calculations are done we can

print out two copies of the statement (one to send in and one to keep) as well as a label. Since we will be printing two copies, I put the statement print routine in a subroutine. (P9 is a tab variable to set how far to the right the column of data is printed. What works best depends upon how long your name and address is.

```
460 P9=32 \GOSUB 620 \GOSUB 620
```

Once the statement is printed, the program switches printers and then prints out the label. The label printer also has to be setup with formatting commands and lines 490-530 do that for my label printer -- a Prism 80).

```
470 PRINT "I'm connecting the printer for labels.",
480 P9="prism"+CHR$(13) \Z=CALL("Prnt",2,0,0,MEM(P9))
490 P9=CHR$(29)+CHR$(6)+CHR$(5)
500 P9=P9+CHR$(27)+"R,19" \REM Letter quality
510 P9=P9+CHR$(27)+"B,89" \REM 6 lines per inch
520 P9=P9+CHR$(27)+"J,0,9009" \REM Set 1 inch margins
530 FOR I=1 TO LEN(P9) \X=ASC(P9,I) \Z=CALL(3124,X) \NEXT
```

Line 540 then prints the label. Because we only want to have the label data in the program once, it is called as a subroutine. The statement printing subroutine also calls this subroutine to print the same data on the statement.

```
540 GOSUB 560 \GOTO 860
```

Line 550 is a subroutine which resets the year to date values at the end of the year.

```
550 Y3=Y0 \Y4=Y1 \Y5=Y2 \M0=1 \Y0=Y0+1 \Y1=0 \Y2=0 \RETURN
```

Line 560 starts the subroutine which prints the name and address of the mortgage company, either on the statement or on the label.

```
560 PRINT:2,""
570 PRINT:2,"Mortgage Company"
580 PRINT:2,"First Address line"
590 PRINT:2,"Street Address line"
600 PRINT:2,"City, State, Zip code"
610 PRINT:2,"" \RETURN
```

Line 620 begins the subroutine to print the statement.

```
620 PRINT:2,TAB(32),"My street address"
630 PRINT:2,TAB(32),"My City, State & Zip code"
640 PRINT:2,TAB(32),M19(M0)," 1,";X41,Y0
650 GOSUB 560
```



```

660 PRINT:2,"Mortgage Account Number 1234567",TAB(P9),
670 PRINT:2,"X#C10F2,B1," Opening Balance"
680 PRINT:2,"Interest Rate      ",%F3,10,"%",TAB(P9),
690 PRINT:2,11," Interest due"
700 PRINT:2,TAB(P9),P1," Principle due"
710 PRINT:2,"First Owners Name  ",TAB(P9),
720 PRINT:2,P0," due ",M09(M0)," 1.",%41,Y0
730 PRINT:2,"Second Owners name ",TAB(P9),
740 PRINT:2,P2," additional principle"
750 PRINT:2,"Owners street address",TAB(P9),
760 PRINT:2,T0," total paid"
770 PRINT:2,"Owners City, state & zip code",TAB(P9),
780 PRINT:2,P3," Principle paid"
790 PRINT:2,TAB(P9),B2," New Balance"
800 PRINT:2,TAB(P9),Y2," Principle YTD"
810 PRINT:2,TAB(P9),Y1," Interest YTD",CHR$(13)
850 PRINT:2,"Insured by:-----a/n 1234567"
820 PRINT:2," \IF Y3=0 THEN 850
830 PRINT:2,"Principle paid during ",%41,Y3," : ",%C910F2,Y5
840 PRINT:2,"Interest paid during ",%41,Y3," : ",%C910F2,Y4
855 PRINT %#, \PAGE:2 \RETURN

```

Once the statements and the label are printed, then we need to store the updated data in the program for the next time it is run. To make darned sure we are at the beginning of the program, we do a rewind. We also want the numbers in the file to be printed with a standard format so the size of the lines will be the same. We only want to replace the first 7 data statements with new ones of exactly the same size, so we set the default print format statement with the %# specification. We will use 3 decimal points of accuracy because the interest percentages are set to the nearest eighth. If your interest rate takes more digits than this the program would have to be modified.

```
860 FILE:4,REW \PRINT %#10F3,
```

Finally, we must write the new data out to the program file, and, just to make sure it's actually written to disk, we close the file.

```

870 PRINT:4,"10 DATA ",B2," \REM Balance      !"
880 PRINT:4,"20 DATA ",10," \REM Interest rate!"
890 PRINT:4,"30 DATA ",P0," \REM Amount due   !"
900 PRINT:4,"40 DATA ",T0," \REM Amount paid  !"
910 PRINT:4,"50 DATA ",Y1," \REM Interest YTD !"
920 PRINT:4,"60 DATA ",Y2," \REM Principle YTD!"
930 PRINT:4,"70 DATA ",Y0+M0/100," \REM Date      !"
940 FILE:4,CLOSE

```

Wouldn't you like to see the result immediately?

```
950 PRINT B2," Balance"
```

```

960 PRINT 10," Interest rate"
970 PRINT P0," Amount due"
980 PRINT T0," Amount paid"
990 PRINT Y1," Interest YTD"
1000 PRINT Y2," Principle YTD"
1010 PRINT Y0+M0/100," Date"

```

And back to Exec.

```
1020 Z=CALL(0,";")
```

If you need more digits, you would need to modify lines 10-70 and line 860 to accommodate the extra digits. The current settings are good enough for mortgages up to 99,999.99. For a bigger mortgage, the above lines would have to be changed accordingly. "10 DATA " is 8 characters long; " \REM Balance !" is 20 characters long; and the format specification is 10F3, so the lines are 38 characters long. Each of the first 7 lines would need to be expanded to fit the larger specification. Here are some examples

```

10F3 - 10 DATA 40000.000 \REM Balance      !
11F3 - 10 DATA 40000.000 \REM Balance      !
11F5 - 10 DATA 40000.00000 \REM Balance      !

```

Now then, would anyone like to see this program converted to PC BASIC?

---

### Advertising

Commercial advertising rates are \$50 for a full page, \$25 for a half page, and \$15 for a quarter page. Anything smaller is \$3.00 per column inch. A column is 3-3/4 inches wide by 10 inches tall. A full page is 7-5/8 inches wide. Non-commercial ads by subscribers are free.

---

Mothers Day vacation in the Berkshires for sale: One week time-share vacation townhouse at Oak and Spruce Resort, Lee, Mass. Sleeps 4, 1-1/2 baths, 19th week (starts on mothers' day). Dues paid to 1994. - Part of a swappable system. - \$3,500. Call 413-354-7750.

---

Wanted to buy — any and all Poly computers. 88, 8810, 8813, twin, 8824; documentation, software, keyboards, spare parts, etc. — Call Charles Steinhauser - Phone: (404) 739-5081 after 7 pm. EST.

---

PolyMorphic 8813 needs home. Make offer. Conway Spitler, P. O. Box 385, Fillmore, CA 93016-0385.

---

PolyMorphic System User Manual, System-88 User's Manual with Exec/96 addendum, & System-88 Operation Essentials On IBM disk. Al Levy, 516-293-8358.

FOR SALE: Poly 8810 box with power supply and mother board. \$50 plus shipping. Charles A. Thompson, 2909 Rosedale Avenue, Dallas, Texas 75205-1532, (214)-368-8223.

**Bit Bucket**

Ground hog day. According to the legend, if the little fellow sees his shadow, we've got six more weeks of winter. Guess what folks? Ground hog day is the middle of winter. How long is winter? Three months. What's half of three months? Yep, you guessed it—six weeks. How long is from the 21st of December—the beginning of Winter—to Ground Hog Day? Just over six weeks. How long is it from Ground hog day to March 21st—the beginning of Spring? A little over 6 weeks. I guess there's a bit of truth in this legend after all.

**DISKS - MODEMS - PROMS - SOFTWARE - SPELL**

1. MAXELL diskettes: 5-1/4" hard sector - \$10 per box.
2. Used diskettes: 5-1/4" 10 hard sector - \$0.50 each.
3. Hayes Micromodem 100 (300 baud S-100 internal modem) - \$20.  
(If you don't have a modem this is a cheap way to go.)
4. HayesSys modem software (for the Micromodem 100) - \$10.
5. Abstract Systems Exec (Enhancements & bugs corrected) - \$30.
6. Abstract Systems Proms (Enhancements & bugs corrected) - \$35.
7. PolyGlot Library \$6 each volume; 5 or more: \$5 each; ALL: \$99
8. Hayes Smartmodem 1200B (IBM compatible internal) - \$30.

Abstract Systems, etc., 191 White Oaks Road, Williamstown, MA 01267, Phone: (413) 458-3597

(Send \$1.00 for a complete catalog--(free with any order).)

(Make check or money order payable to Ralph Kenyon.)

**In This Issue**

Editorial . . . . .	1
Communications . . . . .	1
by Leap(Year)s and Bounds . . . . .	1
Postal Barcodes (part 2) . . . . .	3
Computing your Mortgage . . . . .	7
Advertising . . . . .	9
Bit Bucket (Groundhog Day) . . . . .	10

**PolyLetter**  
191 White Oaks Road  
Williamstown, MA 01267  
(413) 458-3597

Address Correction Requested

**FIRST CLASS MAIL**




Ralph E. Kenyon, Jr.	EXP:99#9
Abstract Systems, etc.	184
191 White Oaks Road	
Williamstown, MA	01267-2256

© Copyright 1991 by Ralph E. Kenyon, Jr.

PolyLetter Editor and Publisher: Ralph Kenyon. Subscriptions: US \$18.00 yr., Canada \$20.00 yr., Overseas \$25.00 yr., payable in US dollars to Ralph Kenyon. Editorial Contributions: Your contributions to this newsletter are always welcome. Articles, suggestions, for articles, or questions you'd like answered are readily accepted. This is your newsletter; please help support it. Non-commercial subscriber adds are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of PolyLetter (1980 through 1990) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 Vol. - \$15, 2 - \$28, 3 - \$40, 4 - \$50, 5 - \$59, 6 - \$67, 7 - \$75; Canada add \$3 shipping, Overseas add \$10. Individual back issues are also available (US: \$3.50, Canada: \$4.00, Overseas: \$5.00).

# PolyLetter



The System-88 Users Newsletter

PolyLetter 92/2

Page 1

MAR/APR 1992

## Editorial

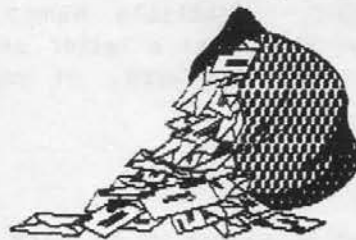
Readers have requested more information on converting from Poly BASIC and the Poly operating system to the PC equivalent. So far I have covered quite a number of such transitions. But I certainly haven't covered all the possibilities. Let's hear from you on specific examples. What problems or difficulties would you like me to try to solve? I'd also like to do more survey-interviews, but I need users to send in their survey sheets. How about it people? Send in some more info on yourselves and your use of poly. (The survey form was printed in *PolyLetter* 91/1.)

## Letters

Hi Ralph,

March 21, 1992

Your last post card reminded me I haven't checked in recently to update the status of my Poly. Up until a few weeks ago everything was business as usual, but then I got a phone call from the other half of the Western NY Poly users group (Doug Schirripa) saying he was moving to California and had no plans to take Poly with him. So now I am the proud(??) owner of two 8813's.



The 'new' system has a habit of wandering off to places unknown, but now that I am hooked on two machines I will probably try [to] fix it and not use the system for spare parts as originally intended. My guess is the memory is infected with Polyheimers disease. When time allows I will swap cards and investigate further. This system also has the CP/M modification. Has anyone had experience with that?

I had seriously entertained the idea of purchasing an IBM clone and sending Poly off

to that great bit bucket in the sky. However with the prices dropping as fast as they are and Poly serving my home needs, what's the rush? Which brings me around to why I haven't renewed my *PolyLetter* subscription. Now that Momma Poly is belly (beak ??) up and the number of users has dwindled, there is not much new information to be had concerning the 8813's. When I do switch to a PC, I plan to make a clean break and not try to create a Poly environment on the PC. So as I review the index for your September/October issue, I don't see anything that grabs my interest. I am not faulting *PolyLetter*, it's just evolution, and some of us are staying behind.

... Also, please keep me on your mailing list, thanks, Ron Moffatt, Rochester, NY.

[We were sorry to hear that Doug left the Poly-Pastures. But we are happy to hear that his Poly has a new home. I visited Doug once, and brought along one of my own Polys. We had difficulty running both Polys on the same electrical circuit. Doug and I concluded that there might have been some kind of interaction between the Polys—electrical noise—that made them act sullen. Each worked fine with the other disconnected. You might check to see if that is the problem.

I have the CP/M modification on some of my machines. It works as well as could be expected (slow). I have had to modify the software in order to use it on my 96-tpi drives. The CP/M modification will allow a Poly to read and write North\* SSSD CP/M disks. I actually bought some software in that format and had it work on the Poly. I bought Nevada COBOL, FORTRAN, and PILOT from Ellis computing. All worked, but I ended up putting them on the shelf. The *PolyLetter* 10-year index (available as PGL-V-33) published in *PolyLetters* 90/4 thru 90/6 lists a number of items about CP/M.

There have been a few readers who disdain the PC world and wish only to know about

Poly stuff. (Just like many disdained the CP/M world). For them, the last issue of PolyLetter contained a Poly program for computing a Mortgage and part 2 of an assembly language program for printing Postal Barcodes. We also wrote a program which computes leap years, and showed how to convert it to GWBASIC; this might have been of interest to both Poly PureHearts and Poly TurnCoats.

Reader requests have been mostly for more information on transitioning to the PC, not necessarily for using Bybee's emulator, but for information on how to convert BASIC programs to run as DOS programs. People aren't trying to create a Poly Environment on DOS machines, (except for a few with huge investments in Poly software); they want to know how to do the same thing their Poly has been doing for them, but in the DOS environment. So far this has meant they want to find a DOS program to do what they did on the Poly, or convert a Poly program to run under DOS—usually GWBASIC. I try to respond to both types of interests (purehearts and turncoats).

I have been running both a Poly and a PC for a number of years now. The Poly still does many things easier than the PC. There are advantages to keeping both running. I have them linked together with a serial cable and transfer files back and forth at 9600 baud. The ease of controlling the Printer on the Poly is still better, in my opinion than, than on the PC. With two minicards installed (port 0 and 1), a text printer hooked up to one, and a label printer hooked up to the other, I can write BASIC programs and command files that switch as necessary. The Mortgage program in the last issue of PolyLetter illustrated this. I now use WordPerfect on the PC as my main word-processor, mainly because, in connection with DrawPerfect, it has superior graphics capabilities. But for many things I still use Poly's Edit and format. The combination of Gnomus and edit allows me to do many things on the Poly that I have not yet figured out how to do on the PC. I did finally figure out how to create a telephone call record and dialer on the PC using the WordPerfect Office Notebook program and a Shell macro. But it is still not as flexible as the combination of Gnomus, edit, Edit, TYPE, and Dial on the Poly. It will be a long while before I abandon the Poly.

By the way, I still have 300 baud modems for the Poly. It was one of these that I used for the dialer program. -- Ed.]

### ***Dimensioning Variables***

In Poly BASIC string variables are presumed to have a fixed size. The default size is the same as DIM O\$(1:8). Any time you want more than that you must explicitly specify the size. GWBASIC string variables, on the other hand, have a variable size. If you add to the variable it takes up more room; if you delete from it, it takes up less room. Don't ask me how GWBASIC does this; I don't know. I know how Poly BASIC does it. Arrays, on the other hand, must be declared in GWBASIC just as they must be declared in Poly BASIC. For example, an array of numbers in Poly BASIC might be declared with the DIM statement as follows:

```
10 DIM A(20)
```

Exactly the same statement works in GWBASIC. Space is set aside for twenty numbers, whose names are A(1), A(2), ... A(20). If we were to dimension a string array in Poly BASIC we must specify not only the number of elements in the array, but the size of the elements. A string array big enough to hold the abbreviations for the names of the months in Poly BASIC would be declared using the statement:

```
10 DIM M$(12:3)
```

In GWBASIC, we would get a syntax error. The corresponding statement in GWBASIC is:

```
10 DIM M$(12)
```

Of course, we can also give the variables longer names in GWBASIC. Variable names in GWBASIC can be longer than just a letter and a digit. We could declare an array of month names in GWBASIC as:

```
10 DIM MONTH$(12)
```

Taking advantage of this allows us to make the program much more readable.

### ***Postal Barcodes (part 3)***

In the first issue we determined that, for my particular printer, I needed to print postnet barcodes using 1 column of dots separated by 3 blank columns, and that the tall bars would use 10 dots while the short bars would use 4. In the second article, we developed the routines to print the dots for one pass.

We needed to make two passes, so we need to look at the routine as we now have it to see how it can be used to print both passes. Here is the routines we have written and explained so far.

```
Stufc STAX D      ;Put in the bar data (in A)
      INX D      ;and move up
      XRA A      ;zero the accumulator
      STAX D     ;Put in one null byte
      INX D     ;bump
      STAX D     ;second null byte
      INX D     ;bump
      STAX D     ;third null byte
      INX D     ;bump for next time.
      RET       ;go back.
```

```
;
; ZipCode digits
Table DB 11000B  ;"0" = 30H
      DB 00011B  ;"1" = 31H
      DB 00101B  ;"2" = 32H
      DB 00110B  ;"3" = 33H
      DB 01001B  ;"4" = 34H
      DB 01010B  ;"5" = 35H
      DB 01100B  ;"6" = 36H
      DB 10001B  ;"7" = 37H
      DB 10010B  ;"8" = 38H
      DB 10100B  ;"9" = 39H
```

```
Xlat  PUSH D      ;Save DE
      ANI 0FH    ;Convert ASCII to number
      MOV E,A    ;Save it here
      MOV A,C    ;Get the checksum
      ADD E     ;add this digit
      DAA      ;Adjust for decimal
      MOV C,A   ;Save the checksum
      MVI A,Table AND 0FFH ;Low address
      ADD E     ;Add the offset
      MOV E,A   ;Replace it
      MVI A,Table/100H ;Hi address
      ACI 0     ;for roll over
      MOV D,A   ;DE now points at
      LDAX D    ;data, so get it
      POP D    ;Restore this
      RET     ;go back
```

```
;Use sign flag to select tall or short bar data
Gd    MVI A,ShortT ;Presume short
      RP          ;Positive so give back short
      MVI A,TallT ;Change to tall
      RET        ;Negative so give back tall
```

```
;Here we process one pass of the zip code; HL points at ZIPcode
DoZc  LDA ZipSiz  ;Get count (5 or 9)
      MOV B,A    ;Initialize counter
      MVI C,0    ;Clear the checksum
SI    MOV A,M     ;Get a zipcode digit
      INX H     ;move up to the next
```

```
CALL Xlat ;Translate it
CALL StufDig ;Stuff bars in buffer
DCR B     ;Count down
JNZ SI    ;More to do so loop
MOV A,C   ;Get checksum dig
ANI 0FH  ;Trim to last Dig ( 7 <= 47)
CMA      ;Convert it to a
INR A    ;negative number (-7 <= 7)
ADI 10   ;add 10 ( 3 <= -7)
CALL Xlat ;translate the checkdigit into bits
StufDig ADD A ;00XXXXX0 <= 000XXXXX
      ADD A ;0XXXXX00 <= 00XXXXX0
      ADD A ;XXXXX000 <= 0XXXXX00
      CALL Stufit
      ADD A ;XXXX0000 <= XXXXX000
      CALL Stufit ;Second bar
      ADD A ;XXX00000 <= XXXX0000
      CALL Stufit ;Third bar
      ADD A ;XX000000 <= XXX00000
      CALL Stufit ;Fourth bar
      ADD A ;X0000000 <= XX000000
Stufit  PUSH PSW ;Fifth & final bar
      CALL Gd   ;Get the bard data (short/tall)
      CALL Stufc ;Put it in the buffer
      POP PSW
      RET
```

```
** HL points at the zipcode buffer **
** DE points at the print buffer **
Graph MVI A,3 ;Turn on graphics
      STAX D
      INX D
      XRA A ;Leading graphics space
      CALL Stufc ;Insert one graphics byte
      ORI 80H ;Set sign to tall for frame bar
      CALL Stufit ;Put in the whole bar
      CALL DoZc ;Insert zip code data
      ORI 80H ;Set sign to tall for frame bar
      CALL Stufit ;Put in the whole bar
      MVI A,3 ;End... †
      STAX D
      INX D
      MVI A,2 ;graphics
      STAX D
      INX D
      MVI A,0EH ;1/2 line feed
      STAX D
      INX D
      RET
```

Just to review, the routine above, which calls the other routines as subroutines, does one of the two passes we need for constructing the data necessary for printing the bar-code. It inserts the necessary graphics control sequence, converts the ZIPcode digits to graphics bars adding the required graphics spaces between bars, computes the POSTNET barcode

checksum, inserts the graphics escape sequence, and finally does a 1/2 line feed in preparation for the next pass of printing.

If you remember, back in *PolyLetter* 92/1, on page 4, we decided what graphics bits to use to print the barcodes. Consider for a moment what is involved in printing the second pass of graphics data. The only difference between the two passes is the actual graphics data used for the upper and lower parts of the tall and short bars. Everything else will be the same. Logically, everything we just did we must do over again, but with different values for the tall and short graphics data. In the first pass we did the top half of the bars. The tall bar was a 7CH and the short bar was a 00H. In the second pass the only difference will be that the tall bars will be 1FH and the short bar will be 1EH. To make things easier to remember, we can create labels for these values. Remember our table?

	Short	Tall
Pass 1	00	7C
Pass 2	1E	1F

Let's give these names and assign the appropriate values. Since these are constants we can set their value for the assembler with EQU statements.

```
ShortT EQU 00H      ;Short Top      |(First
TallT  EQU 7CH      ;Tall Top       |Pass)
ShortB EQU 1EH      ;Short Bottom |(2nd
TallB  EQU 1FH      ;Tall Bottom |pass)
```

I have written the first pass using the labels for the first pass. (What else?) It's really too bad that we can't tell the computer, "Ok, now go back and change ShortT to ShortB and TallT to TallB and do it again." Hmm... Maybe we can—by using an "unsafe programming trick". Here's a perfect opportunity to write "self modifying code". But I am also going to show you how to achieve the same effect without changing the code itself. The difference will save us only four bytes, but the comparison is interesting. Subroutine which actually selects the short or the tall value is "Gd".

```
;Use sign flag to select tall or short bar data
Gd  MVI A,ShortT    ;76 00 Presume short
    RP              ;F0 Positive so give back short
    MVI A,TallT     ;76 7C Change to tall
    RET             ;C9 Negative so give back tall
```

In order for us to change this code, we need to

know where the bytes to be changed are. If we look at the actual code that is generated for this block of data we would see that the location of the value ShortT is one byte past the location of label Gd, and the location for the value TallT is four bytes past the location for the label Gd. The code byte that must be changed for the short byte is located at Gd+1. The code byte that must be changed for the tall byte is located at Gd+4. We can actually change the value after the first pass with something like:

```
MVI A,ShortB      ;Change short for pass 2
STA Gd+1          ;Where it goes
MVI A,TallB       ;Change tall for pass 1
STA Gd+4          ;Where it goes
```

Now, sure as shootin', someone is going to want to "REENTER" this program to save the loading time. That means that the first pass must correct for any code changes the second pass might have done. We need to do the same thing before the first pass, but with the values for the top half of the bars.

```
BarCode MVI A,ShortT ;Set short for pass 1
        STA Gd+1     ;Where it goes
        MVI A,TallT  ;Set tall for pass 1
        STA Gd+4     ;Where it goes
```

Once we have set the right values for the first pass, we must also make sure that HL points at the ZIPCODE buffer and that DE points at the buffer where barcode graphics data is to be created. We assumed that register DE pointed at the graphics data buffer, so we'd better set that up too.

```
LXI D,Barcode    ;Graphics Data area
```

Lets assume that the location of the zipcode to be processed is stored in ZcBuf, a pointer variable. We must point HL at this buffer before each pass.

```
LHLD ZcBuf       ;Our nine digit zip code
```

Now we can call the graphics routine to do the first pass.

```
CALL Graph       ;Do the first pass
```

After the first pass, we must set up for the second pass. We must do almost the same housekeeping for pass 2. We don't have to set up the pointer to the graphics data buffer, or we would replace the stuff we have already put

in it. First we must change the graphic bytes for the Gd routine.

```
MVI A,ShortB ;Change short for pass 2
STA Gd+1 ;Where it goes
MVI A,TallB ;Change tall for pass 2
STA Gd+4 ;Where it goes
```

At this point need to point to the ZipCode buffer again.

```
LHLD ZcBuf
```

We can just put Graph right here; we will have "recursed on our own tail" again—a technique from the artificial intelligence programming language LISP. Here it is all together.

```
BarCode MVI A,ShortT ;Set short for pass 1
        STA Gd+1 ;Where it goes
        MVI A,TallT ;Set tall for pass 1
        STA Gd+4 ;Where it goes
        LHLD ZcBuf ;Pointer to 5 or 9 digit zipcode
        LXI D,Barcode ;Barcode Graphics data area
        CALL Graph ;Do the first pass
        MVI A,ShortB ;Change short for pass 2
        STA Gd+1 ;Where it goes
        MVI A,TallB ;Change tall for pass 1
        STA Gd+4 ;Where it goes
        LHLD ZcBuf

Graph MVI A,3 ;Turn on graphics
      STAX D
      INX D
      XRA A ;Leading graphics space
      CALL Stufc ;Insert one graphics byte
      ORI 80H ;Set sign to tall for frame bar
      CALL Stufit ;Put in the whole bar
      CALL DoZc ;Insert zip code data
      ORI 80H ;Set sign to tall for frame bar
      CALL Stufit ;Put in the whole bar
      MVI A,3 ;End...
      STAX D
      INX D
      MVI A,2 ;graphics
      STAX D
      INX D
      MVI A,0EH ;1/2 line feed
      STAX D
      INX D
      RET
```

I promised to show how this could be done without self-modifying code. One way to do this is to change the routine in Gd from immediate data to indirect data. We would replace the MVI A, instruction with a LDA instruction. The LDA instruction requires two bytes, and we use two of them, so this will increase the code

size by 2 bytes. We also need storage locations for the LDA instructions to get data from. This requires the other two bytes. Here is what Gd would look like.

```
;Use sign flag to select tall or short bar data
Gd LDA Short ;Presume short
   RP ;Positive so give back short
   LDA Tall ;Change to tall
   RET ;Negative so give back tall

Short DS 1 ;Storage for short byte data
Tall DS 1 ;Storage for tall byte data
```

The preparatory routines in BarCode must also be changed.

```
BarCode MVI A,ShortT ;Set short for pass 1
        STA Short ;*** Where it goes
        MVI A,TallT ;Set tall for pass 1
        STA Tall ;*** Where it goes
        LHLD ZcBuf ;Pointer to 5 or 9 digit zipcode
        LXI D,Barcode ;Barcode Graphics data area
        CALL Graph ;Do the first pass
        MVI A,ShortB ;Change short for pass 2
        STA Short ;*** Where it goes
        MVI A,TallB ;Change tall for pass 2
        STA Tall ;*** Where it goes
Graph ... the rest is the same ...
```

So far we have written a routine to build a print buffer full of graphics data to print either a 5 or 9 digit ZIPCode. ZipSize has the size of the ZIPCode, ZcBuf contains a pointer to the text of the ZIPCode, and Barcode is a data area for constructing the print graphics data. The size of that data area must be  $((10 \text{ digits} * 5 \text{ bars per digit}) + 2 \text{ frame bars}) \text{ times } 4 \text{ bytes per bar} + 3 \text{ bytes of graphics control} + 1 \text{ half-line-feed byte}) \text{ times two passes}$ . I get 424 bytes, do you?

```
ZipSize DS 1 ;Size of zipcode - 5 or 9
ZcBuf DS 2 ;Pointer to ZIPCode
Barcode DS (((10*5)+2)*4+3+1)*2 ;Barcode data
```

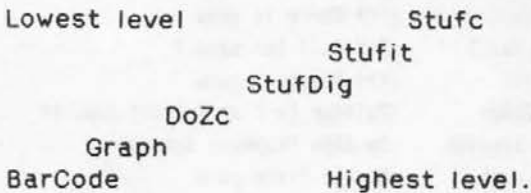
After executing the BarCode routine DE will be pointing one byte past the data in the buffer. We can subtract the start of the buffer from the value in DE; the result will be the number of bytes to send to the printer. Before the program is change to add the barcode graphics routine, we were sending a single carriage return to the printer for a blank line.

There's a lot more to the program to print labels, but there isn't space enough to print the whole program here now. The main routine

I was interested in was to add POSTNET bar-codes to an existing program, and we have covered that.

For other printers, with different graphics, the lion's share of this code would work as is. Only certain graphics commands might need to be changed.

The style of programming illustrated in this article is known as a "bottom-up" processing. We started with a low-level routine and then built a higher routine that used the first one. We did this over and over again until we were "at the top". There were a total of 6 levels, and a few extra routines for good measure.



One certainly can appreciate the ease of programming in a high-level language such as BASIC after sticking this effort out.

Well, that's about enough of this for one issue. I have covered everything necessary for writing a routine to construct POSTNET barcode graphics data for this printer. The rest of the program reads an address list file and prints a label. But that can wait; I will continue that another time. If anyone has other types of graphics printers and would like to see how to adapt this for those printers, feel free to call me.

**Announcement**

**PC Software Adds Postal Bar Codes to Existing Programs**

The Postal Service now allows volume mailers to take a 2-cent discount for pre-sorted, letters and a 1.3-cent discount for post cards that include a nine digit (ZIP + 4) ZIP code and a preprinted postal barcode. The additional preprinted postal barcode will increase the speed and accuracy of mail delivery.

Electronic Technologies introduces POSTBAR, an IBM-PC compatible utility that adds Postal Bar Code capability to your existing word processor, database program, accounting program, mailing program, or custom

programming. POSTBAR is a small memory-resident utility that runs invisibly in the background. No programming is required and there are no program changes necessary to your existing software program. When you print, POSTBAR's artificial intelligence scans the address, locates the ZIP code, and automatically prints a POSTNET bar code above or below the address block.

POSTBAR requires only 12K of memory and will operate on any IBM compatible computer. POSTBAR will print bar codes on any 9 or 24 pin dot matrix printers capable of Epson or IBM Proprinter emulation, PRINTRONIX printers with serial mode, IBM laser printers, and Hewlett Packard compatible laser printers. POSTbar is used by some USPS Business Centers and is USPS certified. The introductory prices for POSTBAR is \$99.

For further information, contact Charles Eglinton, Electronic Technologies, 3985 S. Rochester Road, Suite H, Rochester, MI 48307-5135 or call 313/656-0630

**16K DRAM to 64K DRAM Conversion**

The PolyMorphic 16K dynamic RAM memory boards use 4104 memory chips. Each chip contains 4K by 1 bit of dynamic RAM. The 16K boards use 32 of these chips. There are 4 banks of 8. Each bank is 4K.

When 16K by 1 chips became available the engineers at Poly devised a way to convert the 16K boards into 64K boards. Since Poly does not use the lower 8K, except in the CP/M implementation, the engineers designed the board to support CP/M. The lower 8K is "swapped" in and out for CP/M. The updated board will work in non-CP/M systems because the lower 8K "disappears" in these systems. The new 4116 chips replace the old ones.

I sent a board off to Poly and had it upgraded from 16K to 64K. When I got the board back, I examined it and identified every change the engineers made when they converted it. Since then I have made the change myself to several boards. I am not going to go into the theory of operation of the board, but I will give the steps for upgrading the board. This modification should only be done if your 16K board is working fine. The modification is for Revision D cards only. But the upgrade from Revision C to Revision D is only 4 steps, and I



have included how to do that upgrade as well. If you have Revision D 16K card you can skip the upgrade steps.

In order to do this job, you must know where each IC is located. The diagram on page 8 shows the location of each numbered IC. The view of the board is from the front (component side). When looking at the top of an IC chip, you will see a notch or an indented dot. The

sockets are also usually marked with either a diagonal corner on pin 1 or a notch as in Figure 1. This marks the end of the IC where pin 1 is located. Pin 1 is located to the left, and the pins are numbered counterclockwise. Be very careful when you turn the board over and look at the IC sockets from the other side. The pin locations become the mirror image of when the board is looked at from the front. When looking at the board from the back side, you will see the number 1 pin on the right, and the reversed pins will be numbered clockwise.

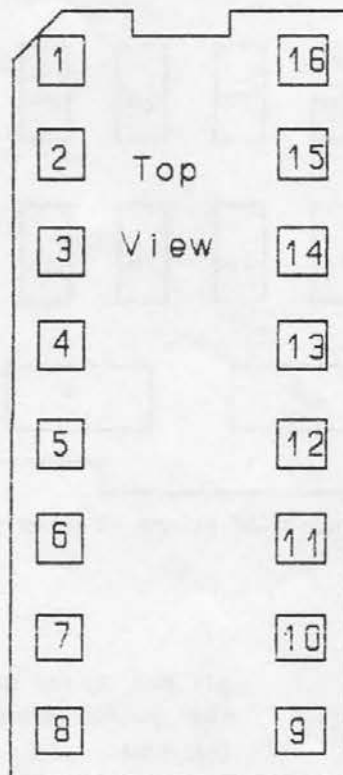


Figure 1

Some IC chips have less than or more than the 16 pin socket in Figure 1. Make sure you always locate pin 1 and count in the right direction, depending upon whether you are looking at the front or back of the board.

PolyMorphic Systems 16K to 64K DRAM modification.

Update REV-C cards to REV-D as follows:

1. Cut trace to IC 1 - 4.
2. Jumper IC 1 - 1 to IC 1 - 6.
3. Jumper IC 1 - 4 to IC 1 - 5.  
See step 4. (See step 10.)
4. Jumper IC 1 - 5 to P1 - 24 (thru feedthru).  
See note 1.

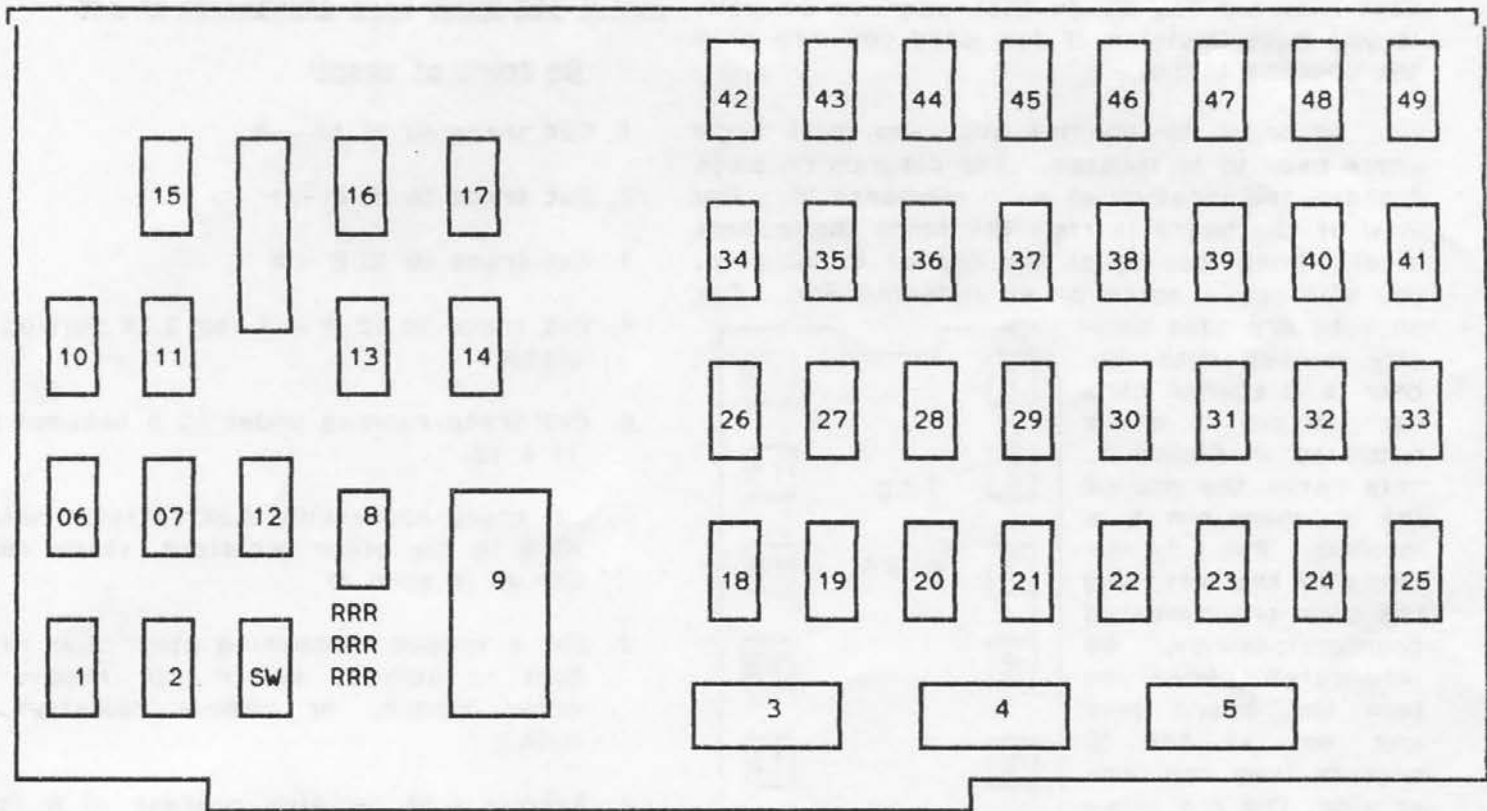
REV-D 16K DRAM card conversion to 64K.

On front of board:

1. Cut trace to IC 17 - 9
2. Cut trace to IC 8 - 1
3. Cut trace to IC 8 - 4
4. Cut trace to IC 8 - 7 (to 2.2K pull-up resistor)
5. Cut trace running under IC 8 between pins 11 & 12.
6. Cut trace connecting 2.2K resistor nearest IC 8 to the other resistors. (Same resistor as in step 4)
7. Cut 4 traces connecting other 2.2K resistors to address switch. (Or remove address switch, or remove resistors, or both.)
8. Remove 2.2K resistor nearest IC 8 (Same resistor as in step 4) and replace with a 39 ohm, 1/4 watt resistor.

On Back of board:

9. Cut trace to IC 7 - 10.
10. Remove jumper, or cut trace, IC 1 - 4 to IC 1 - 5.
11. Jumper left end of 39 ohm resistor (Same resistor as in step 4) to IC 9 - 19.
12. Jumper right end of 39 ohm resistor (Same resistor as in step 4) to the feedthru hole just to the right of IC-18 - 3. (This trace connects thru another feedthru to the trace cut in step 1).
13. Jumper IC 8 - 5 to IC 8 - 8.
14. Jumper IC 8 - 9 to IC 2 - 12.
15. Jumper IC 8 - 12 to IC 12 - 2.
16. Jumper IC 8 - 14 to IC 12 - 3.
17. Jumper IC 10 - 6 to IC 10 - 10.
18. Jumper IC 10 - 8 to IC 1 - 4.
19. Jumper IC 10 - 1 to IC 10 - 2. See step



PolyMorphic Systems 16K dynamic RAM board IC chip positions

20.

20. Jumper IC 10 - 1 to P1 - 16 (thru feed-thru). See note 1.
21. Jumper IC 10 - 3 to IC 8 - 7.
22. Jumper IC 10 - 4 to IC 10 - 5. See step 23.
23. Jumper IC 10 - 4 to P1 - 48 (thru feed-thru). See note 1.
24. Jumper IC 10 - 9 to P1 - 27 (thru feed-thru). See note 1.
25. Jumper IC 9 - 26 to P1 - 33 (thru feed-thru). See note 1.
26. Jumper IC 9 - 27 to IC 8 - 3.

**Last but not least:**

27. Replace the memory chips (IC 18 - IC 49) with 4116's.

**Note:**

1. Solder flows on gold EXTREMELY well; mask

all but 1/16" of the tab on P1 to prevent the solder from flowing onto the rest of the tab.

If you have a working 16K board, but do not want to go to the trouble of upgrading it, I can do the job for a reasonable fee. [413 458-3597]

**Poly Meta**

The Poly Meta Machine was written in 1979 by Lennie Araki. It was Copyright (C), 1979 by PolyMorphic Systems and released to the Public Domain in Mar 1989 by Sirous Parsae.

The Poly Meta machine is a fairly simple machine with only a few structural components to worry about. The machine has a MATCH register which keeps track of the location in the input stream which has been successfully matched. It also has a SWITCH register which reports whether the most recent test instruction had a successful match; its values can be either TRUE, or FALSE.

The following instructions affect the state of

MATCH or SWITCH.

tst 'string' (tests for 'string' as the next characters in the input file.)

SWITCH=TRUE & MATCH points past string, or  
SWITCH=FALSE & MATCH is unchanged.

chr num (Same as tst, but with only one character.)

SWITCH=TRUE & MATCH points past character, or  
SWITCH=FALSE & MATCH is unchanged

cset 'string' (set inclusion; tests to see if the next character is in the given set of characters.)

SWITCH=TRUE & MATCH points past character, or  
SWITCH=FALSE & MATCH is unchanged

char (eat a character)

SWITCH=TRUE & MATCH points past character.

set (set switch to true)

SWITCH=TRUE.

The following instructions are conditional upon the state of SWITCH.

bt addr (Branch if SWITCH=TRUE)  
bf addr (Branch if SWITCH=FALSE)  
be (Branch to Error if SWITCH=FALSE)

Additional control instructions include:

call addr (Call a procedure)  
ret (Return from a procedure)  
b addr (Absolute branch or jump)

Addresses are generated for control instructions by one primitive label generator:

genlb num (Generate a label of the form Lxxxx)

Instructions for inserting text in the output stream include:

outlit string (print literal)  
outchr num (print a character)  
eol (end of line)

Since the Meta machine is used in compiling and translating, primitive instructions for copying text from the input stream to the output file are included.

begst (Mark beginning of star string)

defst num (define star string)

outst num (print star string)

Finally, the Meta machine need to know how to stop and return control to the Poly Operating System.

end (end of program)

The Meta compiler generates a file of Meta machine assembly language instructions, which are handled by the Poly Assembler through the use of MACRO's in the file Machine. This allows the Poly Assembler to assemble code for the non-existent or "virtual" Poly Meta machine. The final result is a native code Poly machine language program.

### Advertising

Wanted to buy — any and all Poly computers. 88, 8810, 8813, twin, 8824; documentation, software, keyboards, spare parts, etc. — Call Charles Steinhauser - Phone: (404) 739-5081 after 7 pm. EST.

PolyMorphic 8813 needs home. Make offer. Conway Spitler, P. O. Box 385, Fillmore, CA 93016-0385.

FOR SALE: Poly 8810 box with power supply and mother board. \$50 plus shipping. Charles A. Thompson, 2909 Rosedale Avenue, Dallas, Texas 75205-1532, (214)-368-8223.

### DISKS - MODEMS - PROMS - SOFTWARE - SPELL

1. MAXELL diskettes: 5-1/4" hard sector - \$10 per box.
2. Used diskettes: 5-1/4" 10 hard sector - \$0.50 each.
3. Hayes Micromodem 100 (300 baud 5-100 internal modem) - \$20. (If you don't have a modem this is a cheap way to go.)
4. HayesSys modem software (for the Micromodem 100) - \$10.
5. Abstract Systems Exec (Enhancements & bugs corrected) - \$30.
6. Abstract Systems Proms (Enhancements & bugs corrected) - \$35.
7. PolyGlot Library \$6 each volume; 5 or more: \$5 each; ALL: \$99
8. Hayes Smartmodem 1200B (IBM compatible internal) - \$30.

Abstract Systems, etc., 191 White Oaks Road, Williamstown, MA 01267, Phone: (413) 458-3597

(Send \$1.00 for a complete catalog--[free with any order].)  
(Make check or money order payable to Ralph Kenyon.)

**Bit Bucket**

Preferred Univocal Notation

It is, as I understand it, a standard expectation that words used in the same context will be used univocally. The axiomatic presumption underlying communicating is that one does not knowingly and intentionally change senses of a term in a single context—except in punning.

Punning is intentionally and consciously, and to make the pun work, sufficiently obviously, shifting senses in mid sentence or paragraph. The difference between punning and equivocating is the difference between consciously or explicitly and unconsciously or tacitly changing the sense of a term in the same context.

In philosophy, mathematics, logic, etc, equivocation is sought out for elimination wherever possible. Con artists, politicians, liars, and other unscrupulous persons do consciously try to dupe the unwary by committing equivocation whenever they can get a favorable reaction in their subjects.

So far, I haven't figured out how to pun in most computer languages. Ada, on the other hand, allows overloading of names—one name can have more than one meaning for a given symbol. Unfortunately, the compiler complains about "ambiguous references" if you try to use an overloaded name in a context where it is not clear which use is intended. Spoil sport!

**In This Issue**

Editorial . . . . .	1
Letters . . . . .	1
Dimensioning Variables . . . . .	2
Postal Barcodes (part 3) . . . . .	2
Announcement . . . . .	6
PC Software Adds Postal Bar Code	
16K DRAM to 64K DRAM Conversion .	6
Poly Meta . . . . .	8
Advertising . . . . .	9
Bit Bucket . . . . .	10
Preferred Univocal Notation	10

**PolyLetter**  
 191 White Oaks Road  
 Williamstown, MA 01267  
 (413) 458-3597

Address Correction Requested

**FIRST CLASS MAIL**




Ralph E. Kenyon, Jr.                      EXP:99#9  
 Abstract Systems, etc.                      184  
 191 White Oaks Road  
 Williamstown, MA                      01267-2256

© Copyright 1992 by Ralph E. Kenyon, Jr.  
 PolyLetter Editor and Publisher: Ralph Kenyon. Subscriptions: US \$18.00 yr., Canada \$20.00 yr., Overseas \$25.00 yr., payable in US dollars to Ralph Kenyon. Editorial Contributions: Your contributions to this newsletter are always welcome. Articles, suggestions, for articles, or questions you'd like answered are readily accepted. This is your newsletter; please help support it. Non-commercial subscriber adds are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* (1980 through 1991) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 Vol. - \$15, 2 - \$28, 3 - \$40, 4 - \$50, 5 - \$59, 6 - \$67, 7 - \$75; Canada add \$3 shipping, Overseas add \$10. Individual back issues are also available (US: \$3.50, Canada: \$4.00, Overseas: \$5.00).

# PolyLetter



## The System-88 Users Newsletter

PolyLetter 92/3

Page 1

MAY/JUN 1992

### Editorial

In the last few issues I have concentrated on Poly related issues. But today there are a couple of PC world items which should be of interest to Poly users who dabble with PCs. In past issues I have let the readers know that I have begun using WordPerfect products to prepare *PolyLetter*. I use WordPerfect 5.1 as my word processor, and I do the graphic images using DrawPerfect 1.1. I also use WordPerfect's Office Notebook package for recording my Telephone Calls. And, lately, I have been doing spread-sheets with PlanPerfect 5.1. Remember Poly's *PLAN* (and Bob Bybee's *Pcalc*)?

Until now the retail cost of these packages has been as expensive now as the Poly was when it first hit the market back in 1976. But WordPerfect Corporation has put together an integrated package which includes scaled down versions of these products for less than a 15th of the combined price. The package is called WordPerfect Works, and the discount price is around \$100. For interested readers there is a more extensive description in the announcements section on page 2.

The second item of possible interest is a hardware upgrade to AT class machines. A new chip/board is available that can turn a 286 machine into a 486 machine! It plugs in place of the CPU chip. That would be about as neat as being able to put a PC class CPU (8088) into a Poly.

Things have been very hectic around here. *PolyLetter* is getting ready to move. The last couple of months have been very busy with getting things fixed up so the house will be ready to put on the market. The housing market is very slow here; property values have declined during the recession. I will be staying here until the house is actually sold. We hope that will be before the next issue is due out around the end of August. But, with today's market,

that may not happen. I will continue to publish *PolyLetter*, and will notify everyone when the time does come. There are houses here which haven't sold in over two years. I sure hope ours isn't one of those. But time will tell. In the meantime, bear with us; I'll keep you posted.

### Letters

Dear Ralph,

June 28, 1992



I recently had a chance to test PM on one of the fastest PCs around, an American Megatrends 80486, 50 MHz motherboard. So here are some new results to add to the bench-

mark list:

<u>system</u>	<u>CPU speed relative to Poly 8813</u>
Poly 8813	1.0
486, 25 MHz	3.6
486, 50 MHz	7.5

So a 486/50 running PM goes 7.5 times as fast as a "real" Poly, in compute-intensive applications which don't involve a lot of disk I/O. This is a little better than double the speed of the 486/25 system which I reported last year. Why not exactly double? Probably due to a difference cache size or architecture on the two systems. I don't have information on the caches used by the two 486 motherboards.

Now there are also 486/66 MHz systems available. These systems use the new Intel "clock doubler" architecture, so the processor uses a 33 MHz clock externally but runs at twice the speed internally. This is the same technology which gave birth to the 486/50 which is really a 486/25 with a doubler

If we assume PM's performance scales linearly with clock speed, and go up from the 25 MHz

system listed above, then PM on a 486/66 would run 9.5 times as fast as a Poly, an 850% speedup. Tasks which took two hours on a Poly would take 12 minutes on PM.

Best regards, and keep on pushing *PolyLetter*.

Bob Bybee, Poly Peripherals, 5011 Brougham Court, Stone Mountain, GA 30087, 404/498-0551.

### Estimating Mortgages

I recently wrote a BASIC program for the PC which estimates the mortgage payment, the closing costs, and the total interest paid. You can specify the interest rate, the percent of the down payment, fees and points, and the term of the mortgage. The program fills in the rest. You would use the program for comparing mortgage rates. Here's a sample output.

Abstract Systems, etc. for Ralph E. Kenyon Jr.	
Loan/Mortgage estimating	
1. Price . . . . .	\$ 80,000.00
2. Percent down . . . . .	10.00
Amount financed . . . . .	\$ 72,000.00
3. Points . . . . .	1.000
4. Fees . . . . .	\$ 1,250.00
The closing costs are \$	9,970.00
5. Interest rate . . . . .	8.000
Monthly payment is . \$	528.31
6. Desired payment is . \$	528.31
The final payment is \$	0.77
7. The term is 30 years.	
The total cost is . . \$	202,132.40
The total interest is \$	118,192.40
Which one would you like to change? _	

### Closing Costs Estimating Program

```

10 COLOR 7,1:READ P,D,P1,F2,C1,T,A:DATA 30000,0,0,0,8,30,0
20 GOSUB 450:GOSUB 410:C1=T:GOSUB 430:CLS
30 F1=(1-1/R1^N)/(R1-1):REM F1=F0/A1 ratio
40 C4=P-F0+P1*F0/100+F2:REM Closing costs
50 A1=INT(F0/F1*100+.99)/100:REM Monthly payment
60 IF A<=F0*R1-F0 THEN A=A1
   :REM test for desired payment too small
70 T1=LOG(A/(F0+A-F0*R1))/LOG(R1):T3=INT(T1+.999999)
80 Y=INT(T3/12):M=INT(T3-Y*12)
90 F4=INT((T3-T1)*A*100+.5)/100 :REM Final payment
100 T4=T3*A+F4+F0*P1/100+F2+C4:REM Total payment
110 I4=INT((T3*A+F4-F0)*100)/100:REM Interest
    
```

```

120 CLS:PRINT
130 PRINT TAB(10);" Abstract Systems, etc."
140 PRINT TAB(10);" for (put your name here)":PRINT
150 PRINT TAB(10);" Loan/Mortgage estimating":PRINT
160 PRINT TAB(10);"1. Price . . . . . ";
   :PRINT USING "$###,###.##";P
170 PRINT TAB(10);"2. Percent down . . . . . ";
   :PRINT USING "#####.##";D*100
180 PRINT TAB(10);" Amount financed . . . . . ";
   :PRINT USING "$###,###.##";F0
190 PRINT TAB(10);"3. Points . . . . . ";
   :PRINT USING "###,###.###";P1
200 PRINT TAB(10);"4. Fees . . . . . ";
   :PRINT USING "$###,###.##";F2
210 PRINT TAB(10);" The closing costs are ";
   :PRINT USING "$###,###.##";C4
220 PRINT TAB(10);"5. Interest rate . . . . . ";
   :PRINT USING "###,###.###";R
230 PRINT TAB(10);" Monthly payment is . . . . . ";
   :PRINT USING "$###,###.##";A1
240 PRINT TAB(10);"6. Desired payment is . . . . . ";
   :PRINT USING "$###,###.##";A
250 PRINT TAB(10);" The final payment is ";
   :PRINT USING "$###,###.##";F4
260 PRINT TAB(10);"7. The term is";Y;"years";
270 IF M=0 THEN PRINT ". ";TAB(79) ELSE PRINT " and";M;"months."
280 PRINT TAB(10);" The total cost is . . . . . ";
   :PRINT USING "$###,###.##";T4
290 PRINT TAB(10);" The total interest is ";
   :PRINT USING "$###,###.##";I4
300 PRINT
310 INPUT " Which one would you like to change?",I
320 IF I=0 THEN PRINT TAB(10);"Thank you, Good bye.":SYSTEM
330 IF I<1 OR I>7 THEN 120
340 INPUT " New value? ",C1
   :ON I GOSUB 370,380,390,400,410,440,430
350 LOCATE 20,1:PRINT TAB(78);" ":PRINT TAB(78);" "
   :PRINT TAB(78);" "
360 PRINT TAB(79):GOTO 30
370 P=C1:GOTO 450
380 D=C1/100:GOTO 450
390 P1=C1:GOTO 460
400 F2=C1:GOTO 460
410 IF C1=0 THEN RETURN
420 R=C1:R1=1+R/1200:GOTO 460
430 T=C1:M=INT(T*12):GOTO 460
440 A=C1:RETURN
450 F0=P*(1-D)
460 A=0:RETURN
    
```

### Announcements

#### WordPerfect Works 1.0

On March 26, 1992, WordPerfect Corporation began shipping WordPerfect Works 1.0, an integrated software package which combines the

LetterPerfect word processor with a graphics editor, database, communications application, and spreadsheet capabilities. The product includes fax capabilities, pull-down menus, mouse support, context-sensitive help and a Run menu to switch easily between programs.

"There is a real need for a powerful integrated package at a reasonable price for home users and traveling business people," said Steve Call, product marketing manager for WordPerfect Works at WordPerfect Corporation. "WordPerfect Works fills that need, is simple to use, and is upward compatible with WordPerfect Corporation's 'older sibling' products."

#### **Word Processor**

LetterPerfect, a streamlined version of WordPerfect 5.1, is the most powerful low-end word processor on the market. Identical file compatibility with WordPerfect 5.1, macro capabilities, graphics integration, a speller, thesaurus and mail merge are offered and more than 900 printers are supported.

#### **Graphics Editor**

The graphics editor allows you to create and edit WordPerfect graphic and text files. Images can be sized, scaled, or rotated, and then inserted into LetterPerfect documents. Files are compatible with DrawPerfect 1.1

#### **Spreadsheet**

With the spreadsheet, you can quickly create and edit charts and graphs; operate more than 100 financial, statistical, mathematical, and logical functions; and import or export Lotus 1-2-3 files. Files are compatible with PlanPerfect 5.1.

#### **Database Management**

You can build custom databases to keep track of important contacts and perform searches, sorts and quick reports for easy printing of mailing lists or merge letters. Three pre-defined databases -- Note Card, Address Book, and Inventory Database -- ship with the product.

#### **Communications**

The communications application in WordPerfect Works, developed by MagicSoft Incorporated for WordPerfect Corporation, allows you to transfer files, download electronic bulletin board service (BBS) files, attach to on-line libraries, and communicate with other individuals through on-line services. A dialing directory and support for modems with speeds from 300 to 38400

BAUD are included. Terminal emulations supported include VT100, VT52, ANSI/BBS, IBM 3101, TTY, and IBM 3270. File transfer protocols include Kermit, Xmodem, Ymodem, and Zmodem.

#### **FAX Capability**

WordPerfect Works adheres to the new FaxBios standard and allows users to send documents anywhere in the world without leaving the program. LetterPerfect creates the fax file and sends it directly to the fax program. The fax image can be examined before transmission in LetterPerfect's View Document feature, then sent to an individual, group, or both.

#### **Integration**

All WordPerfect Works' applications have pull-down menus and context-sensitive help, and are connected by a customizable Run menu. Using the Run menu, you can execute any of the applications from the application you're working in, eliminating the need to go to the Shell menu to switch programs. In addition, if you choose to add additional software programs to the Shell menu, the Run menu will automatically be updated to include the new programs.

WordPerfect Works 1.0 retails for \$159.US/- \$195.CAN and requires 400K free memory, DOS 3.0 or higher, and a hard disk drive. For more information about WordPerfect Works 1.0, call WPCorp's Information Services Department at (800) 451-5151. [Works is available for \$110 from Egghead Software. You may be able to find it discounted even more.]

#### **486 Upgrade for 286**

486 SuperChip™ CPU upgrade for 286 computers offered by Evergreen Technologies, Inc.

Corvallis, Oregon -- April 28, 1992 -- Evergreen Technologies, Inc. announced today the fourth generation of its replacement for the 80286 microprocessor. The Evergreen 486 SuperChip™ CPU upgrade is a faster, high performance version of its predecessor.

Millions of computer users will now be able to inexpensively upgrade their existing 286 machines to run the next generation of performance hungry software such as Microsoft's Windows 3.1 and IBM OS/2 2.0.

The Evergreen 486 SuperChip™ CPU upgrade is a simple replacement for the processor chip in an AT class machine. 286 computer's can be

upgraded to a 486SX simply by replacing the existing 286 processor with an Evergreen™ CPU upgrade. No bus slot is required, and the upgrade works with the computer's existing hardware and software. Replacing the 286 processor with a 486 SuperChip™ CPU upgrade allows AT class machines to run all 386 and 486 specific software, including QEMM386, 386MAX, Microsoft Windows 3.X in 386 enhanced mode, Netware 386, and other 386 programs. The CPU upgrade contains a 486SLC processor and is 100% compatible with all 386 and 486SX specific software.

"The 486 SuperChip CPU upgrade can increase performance three to five times," according to Mike Magee, president of Evergreen Technologies, Inc. "The CPU upgrade is compatible with over one hundred models of 286 computers."

Evergreen Technologies, Inc. first started shipping its original 386 SuperChip™ CPU upgrade in May 1990 for the PLCC socket styles of 286 processors. They have since added PGA and LCC socket styles to their 386 and 486 SuperChip CPU upgrade product line.

"This is the easiest and most cost-effective way to upgrade to a 486 computer," according to Jim Wong, an Oregon computer dealer. "Many of our network customers want to standardize on 386 and 486 machines but can't afford to buy all new hardware. This is a great solution for them."

The list price for the Evergreen 486 SuperChip CPU upgrade is \$399, which does not include math coprocessor support. The math coprocessor model will list for \$499 that includes an on-board math coprocessor. Volume production is scheduled in July with limited quantities available in May and June. The 386 SuperChip CPU upgrade is priced at \$199 and is available now.

Evergreen Technologies, Inc., is located in Corvallis, Oregon and is dedicated to producing CPU and coprocessor upgrades for the computer industry.

For more information please contact:

Mike Magee President Evergreen Technologies, Inc. 915 NW 8th St. Corvallis, OR 97330 (503) 757-0934

## **Poly Meta**

This is the second in a series of articles about the Poly Meta compiler. When Sirous said that I could put the Poly Meta and Orange compilers in the public domain, it was on the basis of the stuff I had received on used disks. Unfortunately, there was no documentation with it. I have studied the compiler and have attempted to understand it well enough to write some documentation. In what follows, there is much guesswork and probably some errors, but it is all I could do without any real experience with the system. Here's what I have found out or guessed by looking at the various parts of the system.

Poly Meta is a high level structured compiler language which allows defining what an input text stream is to look like and may include imbedded output instructions. It appears to have been designed to make writing assemblers and compilers easier, and to facilitate the design of language processors. It's structure is also very useful for translating between languages.

Meta is a batch processor in that it works on an input file and produces an output file. The Meta compiler produces an intermediate code which must be assembled together with another file to produce a working machine language program. This other file is called the 'preamble file', and has some MACROs which define the intermediate code instructions, and some routines which are called by those instructions. Meta is one of those compiler languages that can actually be expressed in itself. Lennie Araki and Larry Deran wrote a version of Meta in itself. The source code I have for PolyMeta is written in PolyMeta itself, and gives some clues to understanding how it works. In discussing what Meta is, I shall be using the source code and discussing what each line does.

Once you have a statement of a language in itself, it can be compiled by itself. In this article I will be showing that compiler in Poly Meta and discussing how to read and understand what the language is doing. There's nothing quite like learning a language by using it at the same time, but there's something quite curious about using it to talk about it at the same time.

Meta in itself is written in a top-down form. The highest level statement is shown first.



Then parts of it are each expanded. In ordinary writing this is like starting with the outline. Then you write the introductory statement, the main theme, and the concluding statement for each section. Then you begin expanding each section as needed. But in writing programs, one often writes routines that are used again and again -- subroutines. But before beginning the program itself I will point out that a comment in Meta is anything inside either paired parenthesis and asterisks "(\*-comment-\*)", or paired slashes and asterisks "/\*-comment-\*/".

What, exactly, a compiler type program is doing is also important for understanding such a program. A compiler is a program that reads a text file of program statements in one language and writes an output file in another language. The simplest form of a compiler is an Assembler. The Poly assembler (Asmb.GO) reads a text file of assembly language instructions and translates that directly into machine readable code. The Poly assembler is a two-pass compiler that does not generate any intermediate files. The first pass builds a symbol table and gets the value of any labels; the second pass rereads the input file and generates the output file, using the symbol table to look up the value of any labels referred to.

Many other compilers actually create intermediate files of various kinds of codes during each pass. Since the Poly Meta machine is actually a virtual machine -- it does not really exist --, the meta compiler produces text versions of the machine instructions I mentioned in *PolyLetter* 92/2. Those instructions will then be assembled using the Poly assembler and a macro library of Meta machine instructions. The Poly Meta compiler is a one-pass compiler, but the job of producing an executable program is finished by the Poly assembler. This means that there will be a total of 3 passes.

Since the compiled Meta code is going to act as an assembly language source code file, it can use symbolic labels, which the assembler will resolve. One neat feature of Poly Meta is that it copies the labels used in the Meta source program for use in the Compiled Meta assembly source list. One advantage to this is to reading the compiled code file. You can see the labels used in the Meta source program and how they fit into the virtual machine code.

Because Poly Meta is a one-pass compiler, the source code for generating output is mixed

right in with the code for recognizing input. We can tell the difference, because code which generates output is enclosed in braces -- "{}". Ok, what's a Meta program?

A Meta PROGRAM is MANY STATEMENTS terminated by a SEMICOLON.

Ok, what does a Meta program look like? It is a sequence of statements followed by a single semicolon. The program may or may not have blank lines or blank spaces before the first statement. But what does a Meta statement look like? A Meta statement consists of an identifier, an equals sign, one alternative form of a statement, and a semicolon, with appropriate spaces or gaps between the various elements. We have almost enough information to describe a Poly Meta Program in a Poly Meta statement, but we need to know how to represent a "sequence of statements". In Poly Meta the dollar sign '\$' is used to denote a repeated structure. We can say that a PROGRAM is a GAP, repeated ('\$') statements, and a semicolon (';'), and when a program is complete, the virtual machine must be told to "end". Here it is in Poly Meta:

```
PROGRAM = GAP $ ST ';' ('[Tab]end');
```

The "repeat" statement ('\$') actually signifies zero or more repetitions. This is a good point to note that the single quotation mark in Meta tells the compiler what it should see next. Anything else would be an error. The characters quote, semicolon, quote, (';') tell the compiler to look for a semicolon as the next character in the input stream.

Remember that this expression of Poly Meta is written in top-down form. We will define what a GAP is and what a ST (statement) is later. Once a program is complete, then the virtual machine must be told that the program is ended. To do that, the Meta definition of a program outputs a TAB character and "end" statement. Because tab characters are invisible, I have replaced them with "[Tab]" in the program so we can see where they are.

We are now in a position to define a statement in Poly Meta. We have abbreviated it above as 'ST', and we have talked briefly about its form. The definition of a PROGRAM is shown in a PolyMeta statement. First there is a string which is an identifier ('ID'). Then there is a tab or space and an equal sign. Following that is another tab or space, one of

the alternative forms of a statement, and a semicolon.

This statement needs to do some house keeping and some outputting too. The first part of the statement is an identifier ('ID') or label, but Meta must remember what that label is to use it in the generated code. Meta must also write that label out. Meta uses a device known as a "star string" to record text strings and copy them out to the object file. An asterisk ('\*' ["star"]) and a digit signifies the compiler to begin recording a star string.

One thing to know about how the Meta compiler works is that each statement in the source language (Meta) generates a subroutine in the object code. Reference to a defined label must create a call to the proper statement. This also means that each compiled statement must generate code that ends with a Meta machine return instruction ('ret').

The definition of a statement must begin by recording a star string that is an identifier ('ID') and then by copying that identifier out to the output file -- "\*1 ID [\*1]". Of course, an identifier must be followed by a GAP, an equals sign, and another GAP -- "GAP '=' GAP". In continuing our top-down defining, the rest of the statement will be an alternative ('ALT') and the terminating semicolon (;). And the compiler will output the virtual machine return instruction. Remember, statements in Poly Meta are treated as subroutines; at the end of a subroutine there must be a return -- '[Tab]ret'. The significance of this is that 'ret' is a meta machine opcode which is implemented as a MACRO which jumps to a routine which clears local procedures and variable off the pseudo machine stack.

What's all this about a pseudo machine stack? Well, the intermediate file consists of macro names of instructions for a virtual or "fictitious" machine [discussed in PolyLetter 92/2]. That machine is a simple one which operates on a stack. A call to a procedure requires allocating stack space for local variables. A return from a procedure requires de-allocating the space. Poly's Assembler compiles (or assembles) assembly source code instructions and outputs 8080 machine code instructions in machine readable form.

Poly's Meta compiler compiles high level source code and output Meta Machine code instructions in assembly source (human) readable form.

These instructions must be further assembled together with another file which defines these instructions. The Meta virtual machine must be emulated by the Poly, and the file Machine contains the assembly source code which allows the Poly to do so.

The Philosophy of Meta is that a statement, or procedure is a callable subroutine. So, every such routine must end in a return to its caller. Hence the definition of ST outputs a Meta Virtual Machine return instruction. So, after the semicolon is found, Meta outputs '[Tab]ret' and a return.

There must also be a GAP between statements, so we can put it at the end of a statement. Here it is all together:

```
ST = *1 ID [*1] GAP '=' GAP ALT ';' '[Tab]ret' GAP;
```

Now we need to define what the alternative forms of a statement are. There are two possibilities. Either we can have a single condition, or we can have a sequence of alternative conditionals connected with a symbol for logical "OR". The symbol is '|'. Since statements "ORed" together form a true statement whenever any one of the statements are true, finding a true statement ends the need to test any more. The compiler needs to generate a branch on true ('bt') to the end of the list of conditionals whenever any one is true. Let us call a conditional 'CON'. An ALT statement must have at least one CON statement, but it could also have repeated ('\$') additional groups of "OR" ('|') and another CON (along with a GAP).

Grouped items are enclosed in parenthesis. Branching to the end of a sequence of conditionals requires a forward jump. To accommodate this the compiler generates a numbered string using the pound sign ('#') and a digit. At the end of the statement the compiler must output the numbered string, which is done by using the brace characters. Here is the whole thing.

```
ALT = CON $ ('|' GAP [ '[Tab]bt[Tab]' #1] CON ) (#1);
```

Now we must define what a conditional ('CON') is, and we still must define what a GAP is.

Remember that the context of Poly Meta is a compiler and that a compiler does two things -- it checks the syntax of the source text and it outputs compiled code. The writers chose to

call the routine that checks out syntax a 'TERM' and the routine that outputs compiled code 'OUTPUT'. So far we have it that a statement in Poly Meta consists of an identifier, an equals sign, and a disjunction of one or more conditions, while a condition can be a sequence of one or more TERMS and/or OUTPUTS.

Let me take a moment here to relate certain aspects of Poly Meta syntax to other languages. An identifier has the effect of testing the input stream for the specified condition, so it is a sort of built-in "if" statement. For example, let's look at the Meta definition of SPACE. SPACE is a definition that only does syntax checking. It does not do any outputting.

```
SPACE = CHR 9 ; CHR 13 ; CHR 32;
```

When the routine SPACE is called, it tests the input stream to see if the next character is a tab, carriage return, or a space.

```
IF
    ASCII(A$,1)=9
OR
    ASCII(A$,1)=13
OR
    ASCII(A$,1)=32
THEN
    SET true
    A$=RIGHT$(A$,LEN(A$)-1)
    REM move up to the next character
ELSE
    SET false
ENDIF
```

For an example of a statement which does both syntax checking and some outputting, let us look at the definition of OUTPUT itself. In Poly Meta an output is set off by braces ('{' and '}'), and one pair of braces can be used to output a sequence of items. Also, a close (right) brace may be preceded by a comma to suppress the trailing carriage return.

```
'}' corresponds to PRINT ""
',}' corresponds to PRINT "",
```

An OUTPUT statement consists of a left brace, a GAP, a sequence of output terms, either a right brace or a comma and a right brace, and a final GAP. In case the output terms are followed by a right brace by itself, it must output an end of line instruction. In Poly Meta:

```
OUTPUT = (' GAP $ OUTER ( '{' ['[Tab]eol' ] ; ',}') GAP;
IF LEFT$(A$,1)="{"
    THEN GOSUB GAP
    (loop) GOSUB OUTER
        IF TRUE THEN GOTO loop
        (keep looping as long as there are outterms)
    ELSE (continue)
ENDIF
IF LEFT$(A$,1)="}" THEN PRINT (an end of line)
ELSEIF LEFT$(A$,2)="," THEN (do nothing)
ELSE (error)
ENDIF
```

Now we can take another look at the CON statement. A CON statement is a sequence of one or more items that either processes a syntax checking TERM or performs an output. In the case the first syntax checking term fails, the routine must skip the rest, so it must branch on false forward to a label at the end of the statement. In the case of the OUTPUT statement, the flag must be set to true. If there are repeated items, then a syntax checking term that fails must generate an error. For the syntax checking only we have a TERM OR an OUTPUT followed by a sequence of either a TERM or an OUTPUT, which looks like:

```
CON = ( TERM ; OUTPUT ) $ ( TERM ; OUTPUT );
```

When we add the output the compiler must generate for the Meta machine we have imbedded output statements including the branch on false, set, and branch on error. We must also have a label to branch to. Here it is:

```
CON = ( TERM [ '[Tab]b[Tab] #1 ] ; OUTPUT ['[Tab]set' ] )
    $ ( TERM [ '[Tab]e' ] ; OUTPUT ) [#1];
```

The largest unit to define is TERM. I won't try to explain all its code in detail, but here is its syntax checking structure.

```
TERM = ( STR
    ; 'CHR' GAP NUM
    ; 'CSET' GAP STR
    ; 'FIND' GAP TERM
    ; ID
    ; '(' GAP ALT ')'
    ; '%' GAP TERM
    ; '*' NUM GAP TERM
    ; '$' GAP TERM
    ) GAP;
```

A TERM is a string, a CHR statement, a CSET statement, a FIND statement, an identifier, an alteration statement, a label statement, a star



```

LETTER = UCLETTER ; LCLETTER;
UCLETTER = CSET 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
LCLETTER = CSET 'abcdefghijklmnopqrstuvwxyz';
DIGIT = CSET '0123456789';
SPACE = CHR 9 ; CHR 13 ; CHR 32;
QUOTE = CHR 39;

```

---

### **Virus Authorship Made Easy**

Virus Research Center  
International Computer Security Association

The U.S. has no laws to deter authors of computer viruses. In July, the "Virus Creation Laboratory" was released through the underground. This highly sophisticated virus authoring package makes it easy for anyone to generate hundreds of new, different viruses.

Earlier this year, the "Little Black Book of Computer Viruses" was published by Mark Ludwig, providing guidance on how to write viruses. For \$15, anyone can order a disk that contains the viruses described in the book. For a small fee he will send you a disk with all the viruses in his book -- a great convenience to virus authors who don't have the time to read his book, and just want to get busy infecting the office. Thus Ludwig also raises the question of whether it should be legal to create or sell computer viruses. A forthcoming book by Mr. Ludwig offers to up the ante by teaching how to write destructive viruses that escape the detection of anti-virus products.

On July 5, a hacker calling himself "Nowhere Man" released version 1.00 of his Virus Creation Laboratory, a slick, professional product intended to write a wide variety of viruses. With this product, you can generate viruses that are encrypted, that resist debuggers, and that can contain up to 10 of 24 preprogrammed "effects" such as clear the screen, cold reboot, corrupt file(s), erase file(s), lock up the computer, drop to ROM Basic, trash a disk, trash some disks, and warm reboot. The viruses are real, have the effects that the program promises, and most are undetectable by today's anti-virus products. Creating a new virus takes just a few minutes with this program. "Products like this are destined to make today's virus problem look like 'the good old days'." says David Stang, Director of Research at the Virus Research Center.

Legislation is needed. Anti-virus prod-

ucts cannot keep pace with the problems created by virus authoring tools and books. Users cannot be protected from viruses by policy, procedure, or training alone. The July issue of *Virus News and Reviews* discusses the Virus Creation Laboratory and the Little Black Book and provides model legislation to deal with the problem. *Virus News and Reviews* is published by the Virus Research Center of the International Computer Security Association, America's only independent virus research and anti-virus product testing lab.

For more information, or to obtain a subscription (#89/year), contact ICSA at 202-364-8252 (Fax: 202-364-1320) or write ICSA, Suite 33, Dept 18, 5435 Connecticut Avenue NW, Washington DC 20015.

---

### **Advertising**

Commercial advertising rates are \$50 for a full page, \$25 for a half page, and \$15 for a quarter page. Anything smaller is \$3.00 per column inch. A column is 3-3/4 inches wide by 10 inches tall. A full page is 7-5/8 inches wide. Non-commercial ads by subscribers are free.

---

Wanted to buy — any and all Poly computers. 88, 8810, 8813, twin, 8824; documentation, software, keyboards, spare parts, etc. — Call Charles Steinhauser - Phone: (404) 739-5081 after 7 pm. EST.

---

PolyMorphic 8813 needs home. Make offer. Conway Spitler, P. O. Box 385, Fillmore, CA 93016-0385.

---

FOR SALE: Poly 8810 box with power supply and mother board. \$50 plus shipping. Charles A. Thompson, 2909 Rosedale Avenue, Dallas, Texas 75205-1532, (214)-368-8223.

---

**DISKS - MODEMS - PROMS - SOFTWARE - SPELL**

1. MAXELL diskettes: 5-1/4" hard sector - \$10 per box.
2. Used diskettes: 5-1/4" 10 hard sector - \$0.50 each.
3. Hayes Micromodem 100 (300 baud S-100 internal modem) - \$20.  
(If you don't have a modem this is a cheap way to go.)
4. HayesSys modem software (for the Micromodem 100) - \$10.
5. Abstract Systems Exec (Enhancements & bugs corrected) - \$30.
6. Abstract Systems Proms (Enhancements & bugs corrected) - \$35.
7. PolyGlot Library \$6 each volume; 5 or more: \$5 each; ALL: \$99  
Abstract Systems, etc., 191 White Oaks Road,  
Williamstown, MA 01267, Phone: (413) 458-3597  
(Send \$1.00 for a complete catalog--[free with any order].)  
(Make check or money order payable to Ralph Kenyon.)

---

**Bit Bucket**

**Polymorph Virus**

by Matt Skoda

Another virus is making itself known on the information BBS. **Polymorph** virus, which is self-encrypting, self-mutating and overwrites random intervals, is presently undetectable except by noticing unexplained **COM** file growth. It spells trouble in PC City. My personal advice is to scan your entire hard drive with McAfee's Virus Scan program with the /AV switch. Then load McAfee's **VShield** TSR virus detection program in your **AUTOEXEC.BAT**. This would notify you of changes to your **COM** or **EXE** files.

Reprinted from the June 1992 ACGNJ NEWS

**Poly Trojans and Viruses**

Was there ever a virus program written for the Poly?. I don't know of one. But I do know of one trojan. (Unfortunately, I wrote it.) The differences: a **trojan**, named after the famous Trojan horse, is a program that appears to do one thing and secretly does another. A

virus is a segment of program code which attaches itself to other executable code and which performs some unwanted function, usually destructive. The pseudo-trojan program I wrote has the real name "Screen-eater.GO" and is a novelty program which blanks the characters from an unattended screen (randomly one at a time). It comes with a Kill-eater.GO program which disconnects the Screen-eater.

**In This Issue**

Editorial . . . . .	1
Letters . . . . .	1
Estimating Mortgages . . . . .	2
Announcements . . . . .	2
WordPerfect Works . . . . .	2
486 Upgrade for 286 . . . . .	3
Poly Meta . . . . .	4
Virus Authorship Made Easy . . . . .	9
Advertising . . . . .	9
Bit Bucket . . . . .	10
Polymorph Virus . . . . .	10
Poly Trojans and Viruses . . . . .	10

**PolyLetter**

191 White Oaks Road  
Williamstown, MA 01267  
(413) 458-3597

Address Correction Requested

**FIRST CLASS MAIL**



Ralph E. Kenyon, Jr.           EXP:99#9  
Abstract Systems, etc.       184  
191 White Oaks Road  
Williamstown, MA           01267-2256

© Copyright 1992 by Ralph E. Kenyon, Jr.

PolyLetter Editor and Publisher: Ralph Kenyon. Subscriptions: US \$18.00 yr., Canada \$20.00 yr., Overseas \$25.00 yr., payable in US dollars to Ralph Kenyon. Editorial Contributions: Your contributions to this newsletter are always welcome. Articles, suggestions, for articles, or questions you'd like answered are readily accepted. This is your newsletter; please help support it. Non-commercial subscriber adds are free of charge. PolyLetter is not affiliated with Polymorphic Systems.

Back volumes of *PolyLetter* (1980 through 1991) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 Vol. - \$15, 2 - \$28, 3 - \$40, 4 - \$50, 5 - \$59, 6 - \$67, 7 - \$75; Canada add \$3 shipping, Overseas add \$10. Individual back issues are also available (US: \$3.50, Canada: \$4.00, Overseas: \$5.00).

# PolyLetter

The System-88 Users Newsletter

PolyLetter 92/4

Page 1

JUL/AUG 1992

## Editorial

I don't have much to editorialize about; I put most of my effort into *Ralph's Ramblings*, but if PolyLetter is to continue to provide Poly Users with good service, some of you will have to send me some ideas to focus on. If you have NOT written or called PolyLetter in the last two years, don't you think it's about time? Call me at 413-458-3597 for a brief chat, and we'll see if we can't get ideas from a conversation. (If you HAVE written or called, write or call anyway.) As Dean Martin used to close his show years (decades?) ago, "Keep them cards and letters comin' folks."

## Letters

Ralph, August 1, 1992

I am still using my Polys daily, but I keep Bybee's PM current on my PC just in case!

### MAJOR CONCERNS:

My Poly keyboards are wearing down! We need to establish & stockpile old keyboards as a source of replacement parts.



I would have had to abandon Poly were it not for my 15 Mbyte hard disk. My 8" disks are impossible to keep aligned! Each drive is different!

I have been working with all the BASICs and Cs on the PC and still am yet to write a useful system that will replace any of my Poly systems. I expect to eventually integrate Microsoft 4.5 with C and assembler (also Microsoft). I tried Bourland's Turbo-C & Assembler first, but I could not integrate it with Basic 4.5.

J. Earl Gilbreath, Savannah, GA

[Earl, One thing that has slowed my conversion of BASIC programs to the PC is that my really useful and interesting BASIC programs use Form.OV. I would have to write all the i/o routines that Form.OV did away with. It's funny you should mention C; I just dug out my old MIX-C compiler and started reading up on it. Remember, we have SMALL-C for the Poly in the PolyGlot library -- PGL-V-11.

Some of us have written assembly language routines which "live" in BASIC variables. Bob Bybee adapted the relocatable loader routine as a machine language overlay to be called by BASIC for loading relocatable files generated by the Poly assembler into BASIC variables. Otherwise, I am not sure how BASIC and MACHINE language programs can be integrated, except as part of command file routines.

In what way were you talking about integrating BASIC, C, and assembler on the PC? How about jotting down your experiences and aspirations as an article for *PolyLetter*? Let's get some conversations about this going! -- Ed.]

## Ralph's Ramblings

**PolyLetter upgrades system!** Well I finally bit the bullet and upgraded my system. Part of my impetus for doing so was the slow speed of my system. To quote a past PolyLetter Editor, Charles Steinhauser, "There's no substitute for clock speed." Well, Charles was right. On one of my many visits to computer shows and flea markets, I had picked up a RAMPAGE expanded memory board for the PC. My old system was an 4.77-8 MHz turbo clone, and I had been used to running it in turbo mode at 8 MHz.

After I installed the RAMPAGE board, I began to get errors in files. After some experimenting, I tracked down the errors to the RAMPAGE board. It looked like I was getting a memory error. So I rummaged around in my box

of parts and found some ram chips. But no matter which ones I changed, I kept getting errors. Finally, I logged onto the AST bulletin board and began reading some stuff about the cards. It looked like the board wouldn't run at 8 MHz except under a PS/2. I couldn't find out anything about the difference, so called the tech support line. They tell me that the RAM-page board was designed to run at 4.77 MHz (IBM PC speed) and won't run at any other speed. Bummer!

Once I found out that the RAMpage board will only run at 4.77 MHz, well, I decided to give it a try. I populated the board with 2 Meg of ram, created a ram-drive and a disk cache to reduce access to the hard drive, and proceeded to test Charles's Dictum. DrawPerfect really slowed down, and WordPerfect's type-ahead buffer began to stack up too many cursor movements for comfort. I'd find that I had pressed too many keys and had to wait for things to catch up. I began to see what Charles meant. On the other hand, both WordPerfect and DrawPerfect ran faster when doing stuff to temporary files, (using the RAM drive), and the cache speeded many things up by caching all the frequently run routines and directories. DrawPerfect does LOTS of numerical calculating, so it was already running slow; cutting the CPU clock speed from 8 to 4.77 made a noticeable difference. Using Office shell to swap programs in and out was much faster, though. That 2 Meg of expanded memory, even at 4.77 MHz, runs lots faster than even a hard disk drive.

Last month I purchased a copy of DataFlex. Al Levy has been programming in DataFlex for quite some time, and he alerted me to a special upgrade offer for a very competitive price. DataFlex wouldn't even install until I freed up 10 Meg of disk space! Boy, I sure miss good ol' Poly's byte miser attitude.

Ok, I finally threw out half the stuff I seldom used and all the stuff I never use and got Dataflex installed. Then I tried to run it. Even with 2 Meg of expanded memory, DataFlex crawled! I'd go over to the pantry and get a snack between menu changes! Talk about slow! I must have gained 10 pounds before I realized what was going on. So, quick, like a Poly, I got on the phone to Al Levy and ordered a 386DX 40Mhz system with a 64K Cache, 1 Meg of RAM, and a 120 Meg hard disk to boot (from) [Get it?]

As Charles would say, "It really smokes!" Now DataFlex only takes about 4 seconds to change menu's. An added bonus is that I can now run the Poly Emulator at a reasonable speed. Of course, I still have lots of programs that can't readily be run on the PC.

### **Two Printers for PM?**

I have been a great one to maximize my use of two printer ports on the Poly. Device 0 is permanently connected to a dedicated label printer. It's an old Integral Data Systems (IDS) Prism-80. Integral Data Systems was acquired by Dataproducts, and that printer is also known as the Dataproducts P-80 series. Anyway, my programs often print out something on the regular printer, switch to the label printer and print out a label. The Mortgage program in *PolyLetter* 92/1 illustrated the technique. It has been darned convenient to use command files that print out things on both printers without my having to touch anything. All that is gone now. I'd like to see if Bob Bybee can upgrade his Poly Emulator so that it understands two printer ports. For Example, Bob's POLYDEV.DAT file allows one to connect the Printer to one of the PC's printer ports. For example:

```
Printer COM2
```

would connect the Poly's output to the PC's second serial port. I'd like to see something like:

```
Printer0 COM4
```

so I can continue to switch printers under program control under the Poly Emulator just like I did on the Poly. When I got my new system from AI, I ordered the extra serial port with switching in mind, but I haven't figure out an elegant way to connect the various printer ports.

Those of you who use PM know that Bob's program PMU allows changing devices on the fly. A command file can change printer ports by calling up PMU. But it is not always easy to execute command files when you are in a BASIC program. Suppose the command file COM2.CD contained the following:

```
PMU
Printer COM2
Q
```

That command file could be accessed by BASIC



in the following manner.

```
99 DIM L$(1:1) \L$=CHR$(13) \REM Carriage return
100 Z=CALL(3) \REM Clear the keyboard buffer
101 OUT 0,"EXEC"+L$+"COM2"+L$+"COM"+L$+"COM"+L$
102 STOP \REM give control back to BASIC
103 REM program continues here
104 REM after COM.CD is done
```

What is happening here is that Line 100 clears the keyboard buffer. If you have earlier versions of BASIC, then Z=CALL(1054) will also flush the keyboard buffer. Once the keyboard buffer is cleared, we use OUT 0 to put some stuff into the buffer, just as if we had typed it ourselves. First we put the command "EXEC" and a carriage return [L\$=CHR\$(13)]. That takes us temporarily out of BASIC and back to Exec. Then we put in the command "COM2" and another carriage return. This runs the command file COM2.CD that we just created. Then we put in the Exec command "CON" and another carriage return. That should bring us back into BASIC. (We should have the double prompt for an interrupted program [>>]). Then we put in the BASIC command "CON" and another carriage return. This should cause BASIC to resume running the program with the statement after the STOP command. Of course, line 101 doesn't actually do all this; it just puts the commands to do it into the keyboard buffer. Those commands will stay there until the STOP statement in line 102 is executed. Then BASIC will stop and the commands in the keyboard buffer will begin executing just as described. When the last command, the second "CON" is executed the program will resume running with line 103.

I have actually done such a thing in my Genealogy program, but I did it to take advantage of frequently used command files for setting up the printers. I have Form.OV on the screen at the time, so I have even figured out a way to turn off the screen display while the command file is being executed. When different printers are selected while under program control, the Poly printer driver software (Prnt.OV) prints its signature on the screen. In my case that signature is:

```
(Printer/{A!S} 20-JAN-84)
```

I shut this off by turning WH1 off. One must be careful, because if an error occurs while the screen is turned off, you may never see it. You will be faced with an apparently dead Poly. Everything it tries to say will be dumped in

stead of put on the screen. Anyway, here's the code I used to shut off and turn back on the screen.

```
2390 X$="prism"+L$ \X9=PEEK(3108) \POKE 3108,201
2400 X=CALL("Prnt",2,0,0,MEM(X$)) \POKE 3108,X9
```

My label printer's name is "prism". By the way, L\$ is defined as follows:

```
DIM L$(1:1)\L$=CHR$(13)
```

Line 2390 sets up for a call to Prnt.OV by putting the printer name and a carriage return in variable X\$. The second statement on line 2390 reads the byte in WH1 and saves it to variable X9. Some systems put a JMP in WH1, and other ones put a CALL there. We can't depend upon it being one or the other, so the best thing to do is to just save whatever was there. The third statement on line 2390 inserts a RET instruction in WH1.

"What's 'WH1'?", I hear some of you cry? I'm glad you asked. 'WH' stands for "Worm Hole". That's the name Poly's original designers gave to what we now call device drivers. Worm hole 0 is the keyboard input driver; worm hole 1 is the video display driver, etc. These two worm holes handle one character at a time. In the case of the screen driver, WH1, it keeps track of the position (POS for you programmers) of the cursor and watches for control characters such as form feed (Clear screen), line feed and carriage returns, delete characters, etc. WH1 is connected to a routine in the PROMS that drives the screen like a simple teletype.

#### **Turning Poly's "ECHO" off (and on)**

Our problem is to shut it off temporarily and then to restore it. The numerical value of the assembly language RET instruction is 201. By **POKE**ing a 201 into the location of WH1, we effectively bypass the display routine. The location (in decimal) is given by Chuck Thompson's addendum for WH1 as 3108. In hex this is... let's see, we keep dividing by 16 until we get a number less than 16.  $3108/16=194.25$ .  $194.25/16=12.140625$ . 12 decimal is a C hex. Since we divide by 16 twice, that means our hexadecimal number must be of the form CXX. Now we must see what the remainder is to compute the next digit. Since we divided by 16 twice, we must multiply 12 times 16 twice to see how much this is.  $12 \cdot 16 \cdot 16 = 3072$ . The remainder is  $3108 - 3072 = 36$ . If we divide this by 16 we get 2.25. That means that the next digit is a 2;

our number is of the form C2X. Now we must subtract the two times sixteen we used; that gives  $36-32=4$ . This means that our hex number is 0C24. If we look in the system programmers guide we see that 0C24 is the value for WH1. If you don't have the system programmers guide, you can still check out the value of WH1 by getting Asmb to look it up in SYSTEM.SY for you. Edit a file called WH1.AS which has the following in it:

```
REFS SYSTEM.SY
REF WH1
END
```

Then you may tell Asmb to assemble the file and to give you a full listing and a symbol table. Here is what you should see:

```
$$Asmb WH1.AS
Macro-88 version 5.1: 06/10/81
Hardcopy? (else video display) (Y or N):N
Full listing? (else errors only) (Y or N):Y
Symbol table printout? (Y or N):Y
Pass one.
Pass two.
```

```
REFS SYSTEM.SY
REF WH1
END
```

Error total = 0

Labels defined in this assembly:

```
WH1      0C24
```

If you want to see all the labels in SYSTEM, then you would change line "REF WH1" to just plain "REF".

Now, let's see, where were we? ... Oh, yes; we were turning ECHO off on the Poly. The third statement on line 2390,

```
POKE 3108,201
```

does the job. Once a 201 is POKEd into 3108, anything sent to WH1 after that will just be discarded. WH1 will be effectively turned off.

Line 2400 calls the Prnt overlay to connect the printer 'prism'; its signature message will not be displayed because WH1 has been turned off. Once Prnt has done its thing, we will want to turn the display back on again. We do that with the second statement on line 2400. We POKE the value that was in WH1 right

back in place. Messages after that will be again displayed on the screen.

Unfortunately, this strategy won't work with the PMU program. It overwrites part of BASIC and the system is likely to crash if BASIC is not restored.

I tried adding "GET BASIC" to the command file, but it apparently wrote over some areas that BASIC keeps as data. BASIC reported "Can't continue!" I took a look at BASIC with Szap and discovered that part of the last sector of BASIC is all zeros. I guessed that this area might be being used for the data that tells BASIC that a program is running and that an error has not occurred.

Poly did it right with Edit.GO. Poly made sure that the data area used by Edit was not in that last sector. We can GET edit at any time without disturbing its data. Too bad BASIC doesn't work the same way.

I decide to "cheat". What I needed is a way to restore the program part of BASIC without disturbing the data area. I reasoned that if I could just reload only the program part of BASIC, things might work. Of course, this wouldn't work with a program that messes with the RAM storage areas used by BASIC. PMU is not that big a program, so, with luck, it might not harm memory past the end of BASIC. OK, sez me, how can I reload only part of BASIC?. My other brain says "It's simple, create a 'short' copy of BASIC. -- Load BASIC and then save as little as you want, but remember to fix it so it can't be run."

How can we do this?. Well, if we look at the ENABLEd LISTing of the directory, we see that BASIC (C04) is 52 sectors in size. We'd like to reload the first 51 sectors. I know! Let's GET BASIC into memory and then SAVE the first 51 sectors. But the SAVE command wants its answers in hexadecimal base. Now we must convert the size of the file to hexadecimal.

Let's see now... decimal numbers use base ten and hexadecimal numbers use base sixteen. In decimal the "tens" digit tells us how many tens we have. 52 sectors is five tens plus two. In hexadecimal, the "sixteens" digit tells us how many sixteens we have. We have to convert five tens plus two into X sixteens plus Y. We can do that by dividing 52 by 16 to see how many sixteens we have. Since three sixteens is 48, it goes three times. 52 minus 48

leaves four left over. So we have three sixteens plus 4 as our hexadecimal number of sectors. 52D sectors is 34H sectors.

Ok, we will want to save one less sector so that that last data sector does not get overwritten. We will have to save three sixteens plus three or 33H sectors. The SAVE command asks for information which tells it where the program being **SAVE**d is. It asks for the From address, the Start address, the Load

address, the number of sectors, and the file name. I started out answering the questions as if they were the same as BASIC.GO

\$\$\$SAVE

From address = 3200

Start address = 3200

Load address = 3200

# of sectors (1-7F) = 33

File name (1 to 31 characters) = BASIC1.GO

**The objective of all dedicated employees should be to thoroughly analyze all situations, anticipate all problems prior to their occurrence, have answers for these problems, and move swiftly to solve these problems when called upon ...**

**However...**

**When you are up to your ass in alligators, it is difficult to remind yourself that your initial objective was to drain the swamp!**

Now we have the problem of preventing the user from running this incomplete version of BASIC, which I shall call BASIC1. I first tried to save it as an overlay. -- That's what it is, you know; it's part of a whole program and that's just what an overlay is. -- "BASIC1.OV" seemed like a good idea to try. That way, if you were to try to run BASIC1.OV, Exec would complain "I don't know what to do with that file." So I deleted BASIC1.OV, **PACK**ed the disk and then **SAVED** it again. Only this time I gave the file name with a .OV extension. (Darn, I could have just **RENAMED** it.)

File name (1 to 31 characters) = BASIC1.OV

When I tried to **GET** BASIC1.OV, Poly complained, rather cryptically, I might add. Wouldn't you know it -- Poly knows too well about overlays. When I tried to use the **GET** command to reload BASIC1.OV in memory on top of the destroyed copy of BASIC in memory, Poly balked. She said: "What?" "What?" is Poly's shortest error message and is used for various subtle stuff. But it isn't very informative. It is also used when some smart-aleck know-it-all tries to do something cute with the system, (like we just tried to do). I **RENAMED** the file back to BASIC1.GO.

I still don't like this; some dummy, namely me, will probably try to run BASIC1 (and probably crash the system). Besides, it's a mark of perfectionism to anticipate this kind of problem and take steps to eliminate it. Speaking of anticipating problems and being ready to fix them, take a look at the box on page 5. (Just thought I'd share that with you.) Ok, ok, back to the problem at hand. How can we prevent BASIC1.GO from being run and still be able to **GET** it? To **GET** it, it must be a runnable program. Let's see, **SAVE** lets you create a runnable program. In doing so it asks for the From address, Start address, Load address, # of sectors (1-7F), and File name (1 to 31 characters). We already tried diddling with the file extension; that didn't work. How about the other parameters? We can't change the From address because that tells **SAVE** where to get the program. How about the Start address? If we could fix it so

that if someone tried to run the program, nothing would happen, that would do the trick. Can we think of some address that won't do anything to the system? The first thing that comes to mind is the address of a **RET** instruction. We could use Szap to poke around in the ROMS to find a location with a 'C9' (the hex value of **RET**) in it and use that. But that is likely to be a different address for different folks with different rom versions.

I know; we'll use the warm start address 0403H. That address doesn't quite do nothing at all; it restores the stack pointer and returns control to Exec. Since Exec already has control in our case, this shouldn't be a problem. We'll try it. Let's go back to **SAVE**

\$\$\$**SAVE**

From address = 3200

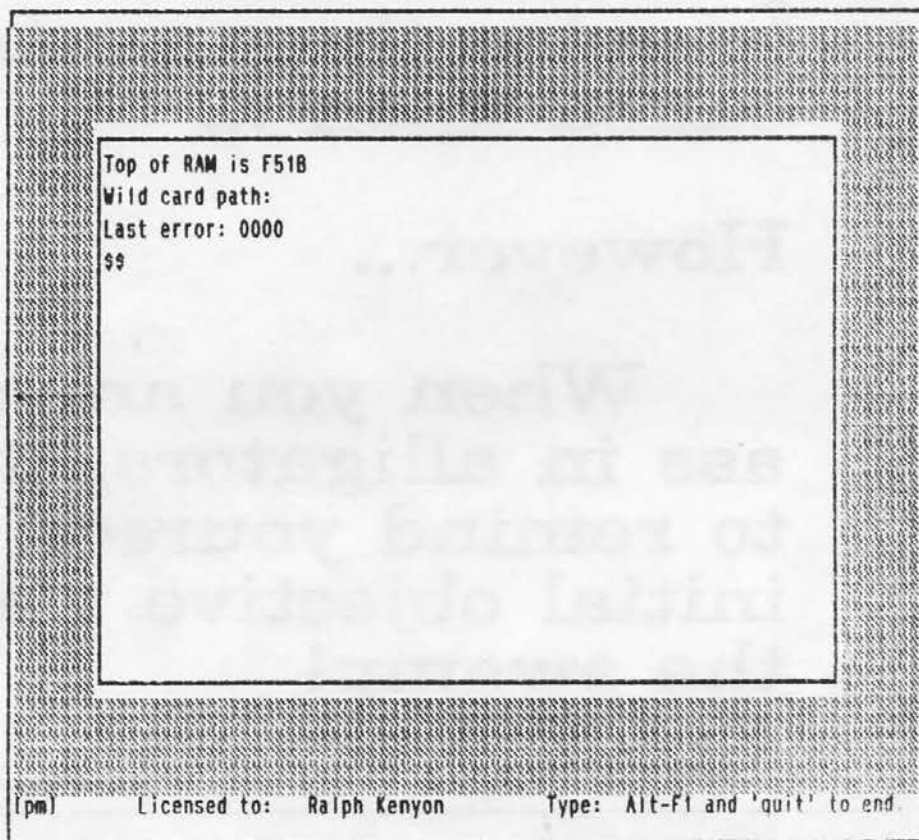
Start address = 0403

Load address = 3200

# of sectors (1-7F) = 33

File name (1 to 31 characters) = BASIC1.GO

Now we can **GET** the file, and we can even try to run it. Trying to run it won't do anything except load it into memory, and that is exactly what we want to do. We could do either of the



The Poly screen centered on the PC.

following in our command file.

1. GET BASIC1.GO
2. BASIC1

Both these would have the same effect; the first 51 sectors of BASIC would be reloaded from disk into memory. It would replace anything trashed by PMU.GO in the low memory area. (Forgot that's what we were doing, did you?). I just hope that PMU doesn't touch anything above 3200+3300=6500. If it doesn't, then we are safe from any other possible trouble (I think). Maybe Bob Bybee can tell us if that's the case with PMU. Better yet, maybe Bob can upgrade PM to include Printer0.

Speaking of upgrading PMU, here's some features I'd personally like to see. My number one desire is to center the Poly Screen on the PC Screen. It would also be nice to be able to have a file called "POLY.SCR" which contained exactly a picture of the 80X25 characters on the screen. This would allow artistic types to draw up a fancy border for the poly screen, one that fits their own idea of how the screen should look.

Of course, if we were really going to concentrate on the esthetics of PM, we would probably want to run the screen in graphics mode, divide the screen into 64 x 16 locations, and use the whole screen for the Poly screen. POP-UP windows could be used for Bob's dialogues with PM itself. Of course, Bob would have to deal with MGA, CGA, EGA, and VGA graphics as well as (perhaps) those few PC users without a graphics adapter. Oh well, we can all dream, can't we?

---

### **Fractal Compression**

PolyLetter has happened onto one of those technology advances that promises to make a real difference. **FracTerm Inc.** is now marketing a system for compressing and displaying images using fractal mathematics. They have a terminal program that allows users to download, decode, and view pictures in a matter of seconds. This development will allow the use of real time on line catalogs for selling just about anything. The secret is in the use of fractal mathematics; this method can be compared to holograms. A small piece of a hologram will show the whole image, but with reduced definition. Fractal compression allows one to achieve a much smaller image file size

while still preserving the overall image.

Let's let them tell their story.

### **Cardz BBS Goes Fractal!**

Cardz BBS is the first bulletin board in the world to offer real-time fractally compressed images. This will allow users to view an image within seconds, using one of the new high speed modems currently available.

**ON-LINE SHOPPING:** Cardz BBS will offer businesses the opportunity to provide pictures of their products along with a text description. End-users will be able to view sports cards, real estate, art, or any other product which needs an image to fully convey its message.

### **FracTerm Communications Software**

**REAL-TIME GRAPHICS:** Our communications program, FracTerm, allows users to access a BBS and display pictures of almost anything in as little as 6 seconds (640x400 resolution and 16.7 million colors). FracTerm also provides multitasking capability which allows a user to view a picture while downloading another at the same time, in the background.

The applications of this software are nearly unlimited. It may be used by a real estate company, for example, to display homes for sale. The user could even search for homes in a certain price range or location.

**WORKS WITH ANY BBS:** Any BBS operator can create an on-line store or image gallery using our FracTerm package. We provide the system operator with all the utilities to catalog and index images. End-users can then access the BBS using our custom software which combines a standard communications program with the added benefit of a graphical interface to view and search fractals.

### **Displaying Fractally Compressed Images (FIF files)**

**STILL IMAGES:** Cardz Computers offers various utilities to view FIF images, either in Windows 3.x, DOS command line or even a full fledged terminal program. Since fractal compression yields such high compression ratios, it is now possible to provide modem users access to not only text but also graphics on any bulletin board system in REAL TIME.

Our FVSmall program can be used to view FIF files on-screen and to convert FIFs to Targa format all from the DOS command line. FVSmall is meant to be used as a shell from within other programs, such as Fox Base Paradox, etc.

**ANIMATION:** Using fractal compression we are able to create color or black and white movies that play at 30 frames/second in a 160 by 100 pixel window. Averaging 250 - 500 bytes per frame we can fit a 2 1/2 minute animated movie on a single 1.44Mb floppy disk including the executable player!!

**GIF FILE PREVIEWS:** With FracTerm BBS users will be able to download fractally compressed previews of GIF libraries very quickly. [This possibility would allow BBS users to quickly see what a given GIF file is a picture of before deciding to download it.] A FIF file is not meant to surpass the quality of a GIF file when being displayed in 8 bit (256 color) mode. However, what it loses in quality it makes up for with its color depth and file size.

### What is Fractal Compression?

**HISTORY:** Fractal Transform technology was first discovered in 1988 by Michael Barnsley. This compression scheme allows the creation of image files, ranging between 10K and 20K bytes in size, while retaining 24 bit (16.7 million) color, and allowing quick transmission over standard phone lines.

**HOW DOES IT WORK?** Fractal compression converts an image into a mathematical representation rather than a series of pixels. While some image quality may be lost in this conversion, we have found that even 10K files provide excellent quality. It is also possible to specify a target file size which controls the resulting image quality of the Fractal Image Format (FIF) file. [Imagine being able to trade off between file size and image definition.]

While the actual compression of images is done with a custom board, using an INTEL i960 RISC CPU and 8 math co-processors, the viewing and decompression of images is done entirely in software and takes as little as two seconds.

### Comparing Fractal Images to Other File Formats

As you can see from table 1, fractal compression provides some amazingly small image

Format	Targa	GIF	FIF
Resolution	640x400	640x400	640x400
Colors	16 million	256	16 million
File Size	768,018	176,764	10,156
Compression Ratio	1:1	4:1	75:1
2400 bps Download	55 min.	13 min.	44 sec.
14.4 Kbps Download	8 min.	2 min.	6 sec.

Table 1: Comparing files CASTLE1.TGA, CASTLE1.GIF, and CASTLE1.FIF

sizes. The impact on graphics transmission over telephone lines will be revolutionary.

If you have any questions, please feel free to contact:

Dwight Jones or John Smith at  
 FracTerm Inc.  
 130-13160 Vanier Place  
 Richmond, B.C.  
 Canana V6V 2J2  
 TEL: (800) 497-3264  
 FAX: (604) 244-7715  
 BBS: (604) 734-5800

Downloads: FracTerm demo - FRACDEMO.EXE;  
 FracTerm terminal program and FIF viewer -  
 FTERM110.EXE

### Help!

In this section I share with you the help system files I have built up over the last few years. (The entire system is included with Abstract Systems Exec.)

#### \$\$\$HELP COMMAND GET

HELP file for system command "GET"

The "GET" command loads a program into memory.

Syntax: "GET [<d>path<file.GO>" (RETURN)

'd' is a drive number and file is an executable program.  
 (see START, REENTER, & ZAP)

"GET [file]" loads a file from the system resident drive.

"GET [<d>file]" loads a 'file' from drive 'd'.

"GET (<d<path<file)" loads a 'file' from drive 'd' and in subdirectory path. "ZGET" combines "ZAP" and "GET".

Minimum size: "G" or "ZG" Example "G <2<GAMES<BREAKOUT"

#### \$\$HELP COMMAND SAVE

HELP file for system command "SAVE"

The "SAVE" command saves a machine program from memory.

Syntax: "SAVE (RETURN) "SAVE" prompts for:

From address = hhhh (hhhh = HEX memory addresses)  
 Start address = hhhh (hh = HEX program page size)  
 Load address = hhhh  
 # of sectors (1-7F) = hh  
 File name (1 to 31 characters) = <d<path<file

Minimum size: "SA"

#### \$\$HELP BASIC CON

HELP file for BASIC CONTROL COMMAND "CON".

"CON" (continue) resumes execution of a BASIC program after a STOP or interruption.

\$\$

HELP file for BASIC CONTROL COMMAND "STOP".

"STOP" halts execution of a BASIC program.

To resume execution of the program after a "STOP" statement, type "CON", "WALK", or "RUN n", where n is the number of the line at which it is desired that execution continue. "CON" and "WALK" continue with the statement immediately following the "STOP" statement at which execution terminated.

### PM Printer Bug Testing

Now that I have a 386DX40, I have been using the Poly Emulator more. One day I tried to **format** a long letter from within PM. I was greatly surprised to discover that some of the printing was garbled. I suspected that something was wrong with the handshaking and began testing the problem. I created a file large enough to reproduce the problem and tried different things. The first thing I did was to try redirecting the output to a DOS file. When I **PRINT**ed that file from DOS, I had no problem. Apparently, PM.EXE misses something when printing to COM ports. Poly's Sio.PM contained input and output buffers which were interrupt driven. In DOS parlance, Sio.PS is a (small) two-way COM port spooler. When Bob replaced Sio.PS to provide the DOS interface, he

changed something and the buffering no longer works correctly.

### BugNote: 43.0 PM Printer

Abstract Systems BugNote 43.0 09/28/92

#### PM Printer Loses Characters.

Bob Bybee's Poly Emulator loses characters when printing to COM ports. When the printer buffer fills up, the Poly Emulator begins to lose characters. As long as you are printing a small file the problem won't show up.

Work around: Redirect the output to a file and use DOS to print the file. There are two ways to do this.

1. Use {AIS} Print-to-a-file (fil.PS) to save the output to a Poly file, copy the file to a DOS file, and then PRINT the file.

2. Use PMU to redirect the output to a DOS file, and then PRINT the file.

### Advertising

Commercial advertising rates are \$50 for a full page, \$25 for a half page, and \$15 for a quarter page. Anything smaller is \$3.00 per column inch. A column is 3-3/4 inches wide by 10 inches tall. A full page is 7-5/8 inches wide. Non-commercial ads by subscribers are free.

Wanted to buy -- any and all Poly computers. 88, 8810, 8813, twin, 8824; documentation, software, keyboards, spare parts, etc. -- Call Charles Steinhauser - Phone: (404) 739-5081 after 7 pm. EST.

PolyMorphic 8813 needs home. Make offer. Conway Spitler, P. O. Box 385, Fillmore, CA 93016-0385.

FOR SALE: Poly 8810 box with power supply and mother board. \$50 plus shipping. Charles A. Thompson, 2909 Rosedale Avenue, Dallas, Texas 75205-1532, (214)-368-8223.

FOR SALE: 3/4 br home. This house nicely combines the features of a 19th century colonial two-story home with a spacious modern addition. A very large carpeted living-room widely open along part of 2 walls to a full-sized kitchen, a full bath, and a double room that cur-

rently serves as a den and the master bedroom make up the downstairs, which is paneled throughout. There is also an enclosed porch, a mud-room, three barns, and an above-ground pool on 3/8th acre of land with nice lilac bush privacy screens from the neighbors. Upstairs are three wall-to-wall carpeted bedrooms and a full bath. The hot water has solar pre-heating, and there is a solar panel on the pool. We're located in the northwest corner of Massachusetts about one mile from Williams College, 5 minutes from Vermont, and 50 minutes from Albany, NY. \$125,000. Call 413-458-3597.

**DISKS - MODEMS - PROMS - SOFTWARE - SPELL**

1. MAXELL diskettes: 5-1/4" hard sector - \$10 per box.
  2. Used diskettes: 5-1/4" 10 hard sector - \$0.50 each.
  3. Hayes Micromodem 100 (300 baud 5-100 internal modem) - \$20.  
(If you don't have a modem this is a cheap way to go.)
  4. HayesSys modem software (for the Micromodem 100) - \$10.
  5. Abstract Systems Exec (Enhancements & bugs corrected) - \$30.
  6. Abstract Systems Proms (Enhancements & bugs corrected) - \$35.
  7. PolyGlot Library \$6 each volume; 5 or more: \$5 each; ALL: \$99
- Abstract Systems, etc., 191 White Oaks Road,  
Williamstown, MA 01267, Phone: (413) 458-3597

(Send \$1.00 for a complete catalog--[free with any order].)  
(Make check or money order payable to Ralph Kenyon.)

**In This Issue**

Editorial . . . . .	1
Letters . . . . .	1
Ralph's Ramblings . . . . .	1
Two Printers for PM? . . . . .	2
Turning Poly's "ECHO" off (and on) . . . . .	3
Fractal Compression . . . . .	7
REAL-TIME GRAPHICS . . . . .	7
Displaying Fractally Compressed Images . . . . .	7
ANIMATION . . . . .	8
Comparing Fractal Images to Other File For- mats . . . . .	8
Help! . . . . .	8
HELP COMMAND GET . . . . .	8
HELP COMMAND SAVE . . . . .	9
PM Printer Bug Testing . . . . .	9
BugNote: 13.0 PM Printer . . . . .	9
Advertising . . . . .	9

**PolyLetter**  
191 White Oaks Road  
Williamstown, MA 01267  
(413) 458-3597

Address Correction Requested



**FIRST CLASS MAIL**

Ralph E. Kenyon, Jr.                   EXP:99#9  
Abstract Systems, etc.                   184  
191 White Oaks Road  
Williamstown, MA                   01267-2256

© Copyright 1992 by Ralph E. Kenyon, Jr.

PolyLetter Editor and Publisher: Ralph Kenyon. Subscriptions: US \$18.00 yr., Canada \$20.00 yr., Overseas \$25.00 yr., payable in US dollars to Ralph Kenyon. Editorial Contributions: Your contributions to this newsletter are always welcome. Articles, suggestions, for articles, or questions you'd like answered are readily accepted. This is your newsletter; please help support it. Non-commercial subscriber adds are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* (1980 through 1991) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 Vol. - \$15, 2 - \$28, 3 - \$40, 4 - \$50, 5 - \$59, 6 - \$67, 7 - \$75; Canada add \$3 shipping, Overseas add \$10. Individual back issues are also available (US: \$3.50, Canada: \$4.00, Overseas: \$5.00).



# PolyLetter



The System-88 Users Newsletter

PolyLetter 92/5

Page 1

SEP/OCT 1992

## Editorial

Well, I've made a slight adjustment to the Masthead. I've been spending a bit of time tracing Mr. Squock in DrawPerfect. The new bird can now be printed at almost any size. I have a large portrait of Mr. Squock on one of my boxes of Poly disks. (It reminds me that that box has Poly disks instead of PC disks.) In case you'd like to do the same, you'll find a similar picture on page 5. If anyone would like the graphics file, give me a call.

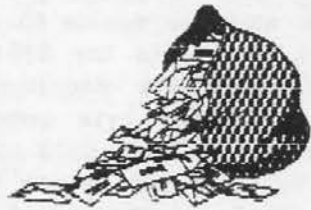
## Letters

Dear Ralph,

October 9, 1992

I'm the Rip Van Winkle of the computer world. My dinosaur equipment is appropriate for the Smithsonian. What I'm saying is that I enjoy my comfortable life with Poly. Most if

not all of *PolyLetter* contents is way over my head. My library of Poly programs (nearly all self made) provides everything I need to manage stock and bond



portfolios and a commodity trading program.

Also, as you know, I have plenty of spare parts to fix 8813's and four different hard disks. Changing to IBM compatible and MS/DOS would be agonizing and traumatizing, I think.

I do have a quest, though. I'd like to buy a laser printer which can be driven by an 8813 with a paper carriage to accommodate paper at least 12 inches wide. There's absolutely nothing wrong at the moment with my Epson MX 100 and my Brother HR-40. But I try to avoid getting caught short or panic situations, so maybe someone out there can tell me what I could get and how to plug it into Poly.

Your dedication to writing *PolyLetter* is simply amazing. It is easy to understand why material to publish is hard to come by. I could write stuff about my use of Poly if I knew specifically what someone wants to know. However to produce a FAX would simplify the process. My FAX number is (513) 531-3106. A sender must insert the document in the machine before dialing to get past the splitter.

Best wishes, Jim Salinger -- Cincinnati, OH.

[Jim, Thanks for writing. You should be able to use any printing device that supports a serial port, including most laser printers. But you will probably not be able to take the maximum advantage of the advanced features of such a device. The standard Poly serial port can only handle 7 bit data. This is because the serial port interface was designed at a time when only 7 bit ascii codes were used in teletype systems. Poly's WH5 driver uses the 8th bit to signal a query the wormhole driver. It can feed back to the Poly information on the status of the device driver.

80H - current line number  
81H - current character position  
82H - lines per page  
83H - characters per line.

I wrote some programs that use this feature. FileSort.GO is the only one I can remember now. I don't know if Poly's WordMaster system uses the feature or not. I also do not know if BASIC uses the feature either. It would be possible to remove this feature by writing a modified version of both Prnt.OV and Sio.PS.

Sio.PS sets up the serial port for 7 data bits, Odd parity, 2 stop bits. It also clears the 8th bit before sending characters to the serial chip (8251A). As long as you have a printer, including a laser printer, on which you can set the communications parameters to Poly's settings, you should be able to, once you get the wiring right, use it with no problems at all.

Earlier issues of *PolyLetter* discussed the wiring options for the header, and for making up cables.

But even if the printer does not have programmable parameters, Sio.PS can be modified to select the proper parameters. I have a Data General Portable printer that uses 8 data bits, no parity, 1 stop bit. I successfully modified Sio.PS to operate with those parameters.

You might think that the job would be difficult, but it's actually not. It just takes a little ingenuity and some knowledge of Sio.PS. Since there are various versions of Sio.PS, I will offer a way to find the data that needs to be changed that does not depend upon the version. The first thing to do is to make a copy of Sio.PS so that we don't wreck the original. Since my printer was a Data General, I decided to name the copy Dio.PS. Since Sio.PS is a system file you can't just use the COPY command. But Scopy will work. If you don't have Scopy, then you can use Reset.GO or Tweak.GO to change the system bit in the directory to a non-system file.

Reset Sio.PS

(Don't forget to change the bit back after copying the file.)

COPY Sio.PS Dio.PS

or

Scopy Sio.PS Dio.PS

Next, you need to use Szap to find and change the appropriate data. If you have RDB (Relocatable Debugger) the job will be much easier. First you need to connect a printer that uses Sio.PS. This will insure that the Sio.PS code is freshly loaded into memory. Then you can start RDB. Once RDB is installed you can bring it up using its hot-key. I think that is CTRL-DELETE. I have modified my version of RDB (rdb) to use CTRL-\ for its hot-key. Anyway, when you press RDB's hot-key you will get something similar to the front panel. RDB gives you everything the front panel does and much more. It can search for code, and that's just what we will use it to do.

**Finding the code to change.** Press the 'S' key. RDB will respond with 'String? '. Type in 'CD AD 2' and press RETURN. (Don't type the quotes.) RDB will then ask 'Starting address? '. Type in '3000' and RETURN. RDB will

then display where it found the code and display the message 'Hit C to continue '. Here's what my screen looks like when I am using Sio.PS from Exec 96.

```
SP OFEE BA FC B9 F5
HL 2D40 0A 20 52 4C
DE 2761 00 20 2D 20
BC 003F E9 D3 08 21
AF 3846 Z (PE)
```

```
0650 LDA 2D88          3008 05 CA 12 30 05 CA 3F 30      0 ?0
0653 ORA A            3010 37 C9 21 00 00 22 95 31  7 ! * 1
0654 JNZ 0678         3018 22 93 31 AF 32 97 31 32      * 1 2 12
0657 DI              3020 98 31 3A E0 31 32 2B 30      1: 12+0
0658 PUSH H          3028 →CD AD 02 1F AA 40 DA 00      e
0659 LHLD 2D84        3030 21 4D 30 22 16 0C 3E 26  !M0* >4
065C LDA 2D82         3038 D3 01 AF 32 B1 2D C9 3E      2 - >
065F CMP L           3040 10 D3 01 AF D3 01 21 64      !d
0660 JZ 064D          3048 00 22 16 0C C9 21 64 00      * !d
```

What is this "CD AD 2" code anyway? 'CD' is the hex value for the assembly language CALL instruction. 'AD 02' is the byte-reversed format used by the 8080 for address 02AD. Together they make up the machine language instruction for CALL 02ADH. 02AD is the address of the routine in the ROMS that sets up the serial port; its name is, would you believe? 'SETUP'. (That's pretty *Smart*, Max.)

SETUP sets up the serial port. The SETUP routine looks for immediate data consisting of one byte for the baud rate and the device followed by non-zero bytes to be fed to the 8251 Universal Synchronous Asynchronous Receiver Transmitter (USART) chip. A zero byte ends the string. In the display above that data is 1F AA 40 DA 00. The '1F' byte selects device 1 and sets the baud rate to 9600. If your printer is slower than 9600 baud, the number will be different. The bytes that follow that program the USART. The first two bytes just make sure that the USART is in programming mode. It's the 'DA' byte that we are going to be interested in. That's the byte that sets up the USART for Poly's parameters. That byte contains bit-by-bit programming instruction for the USART.

The first two bits determine the number of stop bits; the next two bits determine parity; the third pair of bits determine the number of data bits; and the last two bits determine a divide ratio for the system clock. We won't need to change the divide ratio at all, but we can change the others to suit our needs for any serial printer we like. The following chart

is from part of one of my help files and shows the various possibilities.

Parity	Baud Rate Factor	*****
0 Odd	0 0 Sync Mode	* 8251 programming format *
1 Even	0 1 Times 1	*****
\ 0 None	1 0 Times 16	
\ 1 Enable	1 1 Times 64	Sio.PS sets up the serial port for 7 data bits, Odd parity, 2 stop bits.
\ \	//	
X X X X X X X X		
: : 0 0	- 5 bits	
: : 0 1	- 6 bits	Word 1 1 0 1 1 0 1 0 = DA
: : 1 0	- 7 bits	Length
: : 1 1	- 8 bits	FTP sets up the serial port for 8 data bits, Odd parity, 2 stop bits.
0 0	- invalid	
0 1	- 1 stop bits	
1 0	- 1.5 stop bits	
1 1	- 2 stop bits	1 1 0 1 1 1 1 0 = DE

Programming the 8251.

The Data General printer uses only one stop bit, no parity, and 8 data bits. For one stop bit, the first two bits need to be 01; for no parity the next two bits need to be X0, where the X means it doesn't matter; for 8 data bits the third two bits need to be 11; and the last two bits remain the same as 10. This gives us 01X01110. We'll make the X a zero to make it easier. That gives us 01001110 which is 4E hex. To set up Dio.PS for the Data General portable printer, we will need to find the correct DE byte and change it to a 4E byte.

Ok, we've made a copy of Sio.PS and called it Dio.PS. We've used RDB to find the location of the call to SETUP (CD AD 02). Now we must find the programming byte in Dio.PS and change it to the new value. For that purpose we can use Szap.GO. To use Szap, we must first ENABLE the system. That will give us two dollar signs for a prompt, and it will allow Szap to function.

First list the drive with Dio.PS on it and note the value in the Addr column of the listing for Dio.PS. It is a HEX number, so write it down exactly, including any letters. Now we are ready to run Szap.

Szap

You will get a display that looks like the following.

Don't let that cryptic warning scare you. But take heed. Szap is a byte editor and

```

SuperZap version 3.0 02/24/81- Commands are:

^E - Exit      . - Display      ^C - Checksum next 4 sectors
! - Toggle error flag      Z - Zero from cursor
l - Display indirect      :n - Use device n
/n - Display frame n      ' - Enter text
LF - Display prev. frame  CR - Display next frame
→ - Move cursor right    ← - move cursor left
↑ - Move cursor up      ÷ - Move cursor down
nn - Enter data at cursor and move right
ESC - Toggle text display, terminate text entry
nn = Hexadecimal number

If you don't know what you're doing, don't do anything!
    
```

allows editing anything on disk or in memory. We must be very careful to edit only what we want to. Touching any numeric keys or the letter keys 'A' through 'F' will be interpreted by Szap as editing. Accidental editing in the wrong place may result in destroying a system program or the directory itself.

First we press the colon followed by the drive Dio.PS is on and RETURN. Next we press the forward slash ('/') followed by the value in the Addr column and RETURN. The upper right corner should contain the drive number followed by the Addr value. If you are using Exec/96, your display should look like the following, but the address in the upper right corner may be different.

```

5 CA BF 30 05 CA 49 31 05 CA 12 30 05 CA 3F 30
37 C9 21 00 00 22 95 31 22 93 31 AF 32 97 31 32
98 31 3A E0 31 32 2B 30 CD AD 02 00 AA 40 DA 00
21 4D 30 22 16 0C 3E 26 D3 01 AF 32 B1 2D C9 3E
10 D3 01 AF D3 01 21 64 00 22 16 0C C9 21 64 00
E5 DB 01 1F DA 7E 30 1F D0 DB 00 E6 7F 4F 3A EA
31 B9 C2 6E 30 AF 32 98 31 3A E6 31 B7 C0 11 99
31 06 05 2A 95 31 CD 6C 31 C8 22 95 31 C9 E6 40
CA AB 30 11 9E 31 06 42 2A 93 31 CD 7D 31 CA 97
30 22 93 31 D3 00 C9 DB 01 E6 04 CA 97 30 3E 36
D3 01 AF 32 B1 2D D3 00 D3 00 C9 CD 97 30 FB 76
DB 01 17 D2 AE 30 F3 3E 27 D3 01 32 B1 2D C9 21
E1 31 23 FE 0D CA 5D 31 23 FE 0A CA 5D 31 23 FE
09 CA 5D 31 23 FE 08 CA 5D 31 E6 7F 4F 21 B1 2D
7E B7 C2 EB 30 F3 3E 37 D3 01 77 3A E6 31 B7 CA
2A 31 5F 79 FE 1B 06 01 C2 FD 30 06 03 3A 97 31
    
```

I have marked the code we are looking for in BOLD on the display. Notice that the byte following CD 02 AD is a 00 byte instead of 1F or whatever you found. That is because the baud rate byte is only set after Prnt.OV in-

stalls the driver. Setup.GO stores the answers to your questions in Prnt.OV. When you select "Printer [name]", Prnt.OV looks that data and inserts it in the proper place in memory. Poly might have made Setup, Prnt, and Sio, jointly programmable with the questions about baud rate, parity, and stop bits, but I see two reasons why not. One is that printers began to have switches to set these parameters. The other is that these parameters might have been perceived of as "too technical" for even the 1976 users. (The installing technician had to do some work.)

This change only corrects for the serial data parameters; it does not allow printing an 8 bit character to the printer.

### Printing 8 bits in PH.EXE

The Genealogy program I talked about a few issues back uses extended ascii characters to draw a family tree, and I wanted to be able to send that 8th bit to a file for printing in WordPerfect. I took a look at what WH7 (and WH5) does to characters when it processes them. I found out how to turn the 8th bit on, at least when printing to a file. The Worm Hole driver is part of Prnt.OV and gets copied into the region from 2F00 to 2FFF whenever a defined printer is selected.

I looked at that code with the idea that I would use BASIC to POKE some changes into it. It turned out that I could do the job with only two POKE's. Let's look at the code involved. I have marked the two changes needed.

```

ORG DrvLoc
LOC WhDrv      ;Worm holes WH5, WH6, WH7 driver code

WH5  PUSH B      ;WH5 driver code - bypass pagination
     PUSH D
     PUSH H
     CALL ChOut
     POP H
     POP D
     POP B
     RET

WH6r  PUSH B      ;WH6 driver code - get char
      PUSH D
      PUSH H
      MVI B,2
      CALL GetCh
      POP H
      POP D

```

```

POP B
RET

WH7r  ANI 07FH      ;<-- Change to ANI 0FFH --POKE 12055,255
      PUSH PSW      ;WH7 code - character out to printer
      PUSH B
      PUSH D
      PUSH H
      LXI H,loret
      PUSH H
      IF EQ,VT,ChOut ;CHR$(11)
      IF EQ,FF,ChOut ;CHR$(12)
      PUSH PSW
      LXI H,lpos
      LDA bottom
      CMP M
      MVI A,FF      ;Do a form feed at end of page
      CC ChOut
      TMr  LXI H,top
           ck lpos,GE,M,$+9
           CALL LFOut ;Do top margin
           JMP TMr
           ;
           POP PSW
           IF NE,CR,ChOut
           CALL ChOut
           LFOut MVI A,LF ;Automatic Line feed routine
           ChOut MOV C,A
                ORA A
                JP ChOut1 ;<-- Change to JMP--POKE 12116,195
                MVI H,lpos/100h ;8X returns parameter info
                ADI lpos AND 7FH
                MOV L,A
                MOV A,M
                RET

           ChOut1 SUI TAB ;Check for control characters
                  JZ TABr
                  DCR A
                  JZ LFr
                  DCR A
                  JZ VTr
                  DCR A
                  JZ FFr
                  DCR A
                  JNZ Print
                  CRr  CALL PutCh
                       set cpos
                       LXI H,Edge
                       MOV M,A
                       ck devtyp,NE,0
                       LDA edge
                       MOV M,A
                       RET

           LFr  CALL PutCh
                LXI H,lpos
                INR M

```

```

ck lpp,NE,M
MVI M,0
RET

TABr MVI A,Space ;JMP uTABr
CALL ChOut ;This gets modified if the printer
LDA cpos ;understands tabs
ANI ?
JNZ TABr
RET

uTABr CALL Print
LXI H,cpos
MOV A,M
ADI 8
ANI 0FBH
MOV M,A

RET

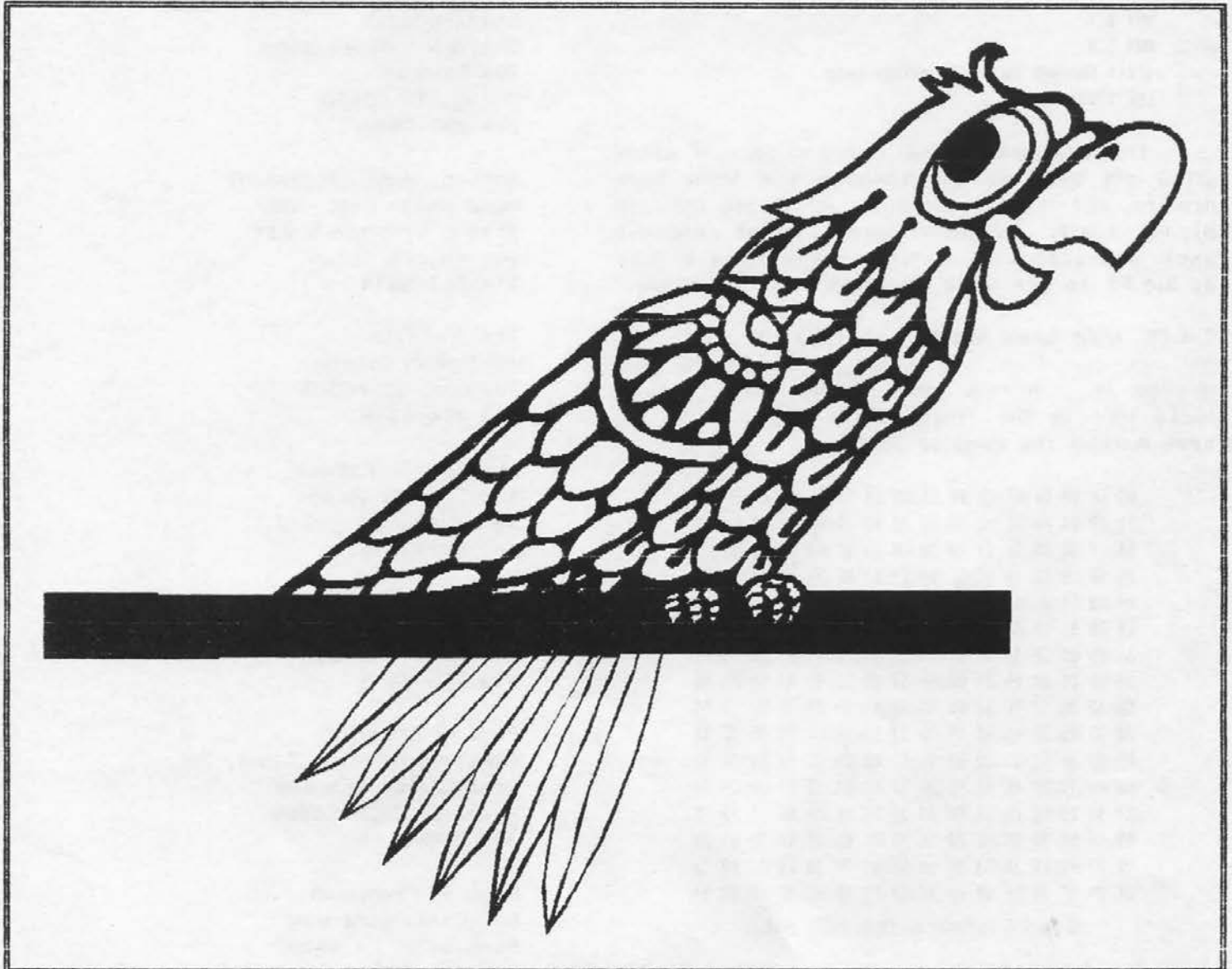
VTr ck lpos,EQ,0
INR C ;Change to FF
FFr CALL LFOut ;JMP uFFr
ck lpos,NE,0,FFr ;This gets modified if the
RET ;printer understands formfeeds

uFFr CALL PutCh
set lpos
RET

Edge DS 1
DS 3

Print LXI H,Edge ;Left edge routine
MOV A,M

```



"Mr. Squock"

```

IF EQ,0,Print1
DCR M
PUSH B
MVI C,Space
CALL PutCh
POP B
JMP Print

```

```

Print1 LXI H,cpos
MOV A,C
IF NE,BS,$+4
DCR M ;Back space counts back
;
IF LT,Space,Print2 ;Skip control
INR A ;7FH becomes 80H
JM Print2 ;Skip DEL
INR M ;Count the char
Print2 ckl cpl,LT,M ;lpos > cpl
PutCh MOV A,C ;Get the char
MVI B,1
GetCh MVI C,0
;Fall through to Sio.PS driver code
LOC 2700H

```

The changes in the above code will allow an 8 bit byte to get through the Worm hole drivers, but it will not allow it to get through Sio.PS itself. My desire was to print extended ascii characters to a file. Let's take a look at Sio.PS to see what would be required there.

Sio.PS also uses ANI 7FH to clear the 8th bit. But we can find the location in Sio.PS and change it. Here's what Sio.PS for Exec/96 looks like in the first sector using Szap. I have marked the code in bold.

```

05 CA BF 30 05 CA 49 31 05 CA 12 30 05 CA 3F 30
37 C9 21 00 00 22 95 31 22 93 31 AF 32 97 31 32
98 31 3A E0 31 32 2B 30 CD AD 02 00 AA 40 DA 00
21 4D 30 22 16 0C 3E 26 D3 01 AF 32 B1 2D C9 3E
10 D3 01 AF D3 01 21 64 00 22 16 0C C9 21 64 00
E5 D8 01 1F DA 7E 30 1F D0 DB 00 E6 7F 4F 3A EA
31 B9 C2 6E 30 AF 32 98 31 3A E6 31 B7 C0 11 99
31 06 05 2A 95 31 CD 6C 31 C8 22 95 31 C9 E6 40
CA AB 30 11 9E 31 06 42 2A 93 31 CD 7D 31 CA 97
30 22 93 31 D3 00 C9 DB 01 E6 04 CA 97 30 3E 36
D3 01 AF 32 B1 2D D3 00 D3 00 C9 CD 97 30 FB 76
DB 01 17 D2 AE 30 F3 3E 27 D3 01 32 B1 2D C9 21
E1 31 23 FE 0D CA 5D 31 23 FE 0A CA 5D 31 23 FE
09 CA 5D 31 23 FE 08 CA 5D 31 E6 7F 4F 21 B1 2D
7E B7 C2 EB 30 F3 3E 37 D3 01 77 3A E6 31 B7 CA
2A 31 5F 79 FE 1B 06 01 C2 FD 30 06 03 3A 97 31

```

Sio.PS clears the 8th bit.

The bytes E7 7F is the code for ANI 7FH. That code appears twice in Sio.PS, but the one we need to change is followed by 4F [MOV C,A].

If we change the byte 7F to FF that will prevent clearing the 8th bit. The address of the 7F byte for Exec/96 is 30DBH, which is 12507 decimal. We can use BASIC to POKE,12507,255. This should allow sending 8-bit bytes through Sio.PS. The addresses for other Exec versions may be different, so we should use RDB or Szap to find the correct addresses.

### *Poly's for sale or donation*

Since the last issue I have taken a look at my file of recent responses from former Poly users. I have gleaned from those responses a list of people who wish to sell or donate their Polys. These Polys can be thought of as a repository of spare parts. Here they are

Charles Mach  
Charles & Helen Mach  
709 Bowman  
Irving, TX 75060  
214 259-7964

Anthony Muse, President  
Muse Metal Lab., Inc.  
1110 S Virginia B 831  
Terrell, TX 75160  
214 563-5614

Tim Ziebarth  
4641 Huey Circle  
Boulder, CO 80303  
303 494-8579

Stephen R. Krause  
111 Croydon Road  
Baltimore, MD 21212  
301 532-9079

John Strickland, Jr.  
12717 Bullick Hollow Road  
Austin, TX 78726  
512 258-8998

Paul H. Emerling  
Emerling Machine Tools, Inc.  
Post Office Box 626  
Nederland, CO 80466  
303 258-3556

Hugh F. Frohbach  
573 Connemara Way  
Sunnyvale, CA 94087  
408 736-9324

John W. Wrenn  
Route 5, Box 944  
Abilene, TX 79605

Graham Mullins  
Mullins Computer Service  
Post Office Box 11423  
Montgomery, AL 36111  
205 288-7059

Jill Reed  
Burton/Hawks, Inc.  
Post Office Box 359  
Casper, WY 82602  
307 234-1593

Robert A Teall  
Bob's Industrial Electronics  
1720 North Sherman Avenue  
North Platte, NE 69101-7299  
308 532-8017

Vic Taylor  
Taylor Consulting  
N 17016 Madison Road  
Mead, WA 99021  
509 238-4413

If you get one of them please let me know so I can clear the name from my list.

---

### ***Ralph's Ramblings (more)***

Backing up the hard disk on my Poly is something I have gotten down to a science. I put the backup disk in drive 2, and then I press the BACKSPACE key followed by 'b' (BS, b). Sometimes I get the message "Output disk is full... Insert new disk and press any key." But when that happens I usually hit Ctrl-Y, pack the backup disk, and then press BACKSPACE, b again. On occasion, I get the message "I can't find that file." Then I know I haven't set up a command file on the backup disk. Then I press BACKSPACES, B, supply the name of the directory to be backed up, and press RETURN. This creates the backup command file on drive 2. Then I can press BACKSPACES, b, and the directory will be backed up as usual.

Sounds easy to do, right?. Well, a lot of effort went into making this work easy, and I am still trying to figure out how to do something similar on the PC. Let's take a look at what all goes into this seemingly simple operation. First, what's all this BACKSPACE, b

stuff? Recall the article *Two Poly TSR's: Gnomus with edit*, back in *PolyLetter* 88/5? Gnomus is a TSR (Terminate and Stay Resident) macro utility. A PC program which accomplishes a similar effect is ProKey. Prokey works by watching the keyboard. It has a list of definitions for keys, and watches for those keys being pressed. If one of those keys is pressed, ProKey takes over and substitutes its definition for the selected key. Because the PC has 3 different shift keys (Shift, Ctrl, & Alt) many different combinations may be used. You could replace Alt-D with "DIR,ENTER". This would allow you to get a directory listing just by pressing Alt-D.

The Poly keyboard generates ASCII characters for all its shift key combinations, and the Poly PROMS process keyboard interrupts just by putting the character pressed into a buffer. Exec, or some running program gets characters from this buffer. Things work a bit differently on the PC. The PC PROMS store something called the "scan code" which tells which key was active and whether it was pressed or released. This scan code information is processed along with the status of the various shift keys and both the scan code and the character value are put in the buffer. A smart PC program can keep track of the various shift keys by them selves as well as the ASCII value for the key pressed. ProKey does this so we can define Alt-keys or Ctrl-keys to be "smart" (Incidentally, there's a similar program called "SmartKey", but since I have ProKey, I'll talk about ProKey.) Many PC programs also have built in Keyboard Macro capabilities. WordPerfect programs are examples.

But the Poly PROMS only put the ASCII value of the character in the buffer. For Gnomus to work, one of these keys must be set aside to perform the "Hey, Gnomus!! The next key is for you!!" function. Gnomus eventually became the Poly Editor Escape library facility, so Poly originally used the ESC key to activate it. When I upgraded Gnomus for use outside the Editor, I wanted a different key. Since the Poly system almost never uses the BACKSPACE key, I chose it. Now, when Gnomus is installed, it watches for a BACKSPACE character. When Gnomus sees a BACKSPACE, it takes the next character as one for it to replace with a macro.

Of course, we might actually want to type a backspace character, so Gnomus knows that a second backspace is to be passed on for

other programs. Anyway, I told Gnomus to replace the 'b' character with the string '<2<bu.CD', and the RETURN key. I do this by pressing BACKSPACE,=. Gnomus then records every keystroke until it sees ESC, RETURN (unless its buffer fills up first). Defining 'b' used the following keystrokes:

```
BACKSPACE, =, b, <, 2, <, b, u, ., C, D, RE-
TURN, ESC, RETURN
```

After the macro is defined I can use it. By pressing BACKSPACE, b; I actually get <2<bu.CD, RETURN, as if I had typed it myself. Basically, Gnomus feeds its replacement characters to any program asking for characters from Worm Hole zero (WH0). Then net result is that when I hit BACKSPACE, b, Gnomus tells Exec to executed the command file "bu.CD" on drive 2. Every backup disk is different, but the command file for backing up has the same name. It sure makes life easier -- at least it did till I tried to do the same thing on the PC. (For more details on this see *File Management* in *PolyLetter* 91/4.) On the Poly, BACKUP.GO looks at files in its directory to see if the new bit is set. Poly's system sets the new bit on files when they are created or modified by Edit or BASIC. BACKUP.GO copies all files with the new bit set and may be told to replace the files. For example, my backup disk for directory GS on drive 7 has the following in it.

```
U<BACKUP 7<GS 2 *
```

The asterisk tells BACKUP to replace files with the same name.

On the PC the archive bit performs a function similar to the new bit on the Poly. In later versions of DOS, XCOPY.EXE can be made to work similar to Poly's BACKUP.GO, but we must explicitly tell it to copy only modified files. We do that by using the "/M" switch. Poly's BACKUP.GO automatically copies all sub-directories. When we use XCOPY in DOS, we must also tell XCOPY to copy subdirectories by using the "/S" switch. Here's what the command would look like in DOS.

```
C:\DOS\XCOPY C:\GS\*. * E:\GS /M /S
```

There are some other subtle differences that we should know about and remember. On the Poly, the RENAME command automatically clears

the new bit. But on the PC, the RENAME does not change the archive bit. Also, on the Poly, BACKUP.GO does not automatically replace existing files unless the asterisk is supplied. On the PC replacement is the default option. Here's MSDOS 5.0 response to HELP XCOPY.

```
Microsoft(R) MS-DOS(R) Version 5.00
(C)Copyright Microsoft Corp 1981-1991.

$$HELP XCOPY
Copies files (except hidden and system files) and directory trees.

XCOPY source [destination] [/A : /M] [/D:date] [/P] [/S [/E]] [/V] [/W]

source      Specifies the file(s) to copy.
destination Specifies the location and/or name of new files.
/A          Copies files with the archive attribute set,
           doesn't change the attribute.
/M          Copies files with the archive attribute set,
           turns off the archive attribute.
/D:date     Copies files changed on or after the specified date.
/P          Prompts you before creating each destination file.
/S          Copies directories and subdirectories except empty ones.
/E          Copies any subdirectories, even if empty.
/V          Verifies each new file.
/W          Prompts you to press a key before copying.
```

"edit" is screen editor which outputs to the command buffer. A PC program which accomplishes a similar effect is DOS-EDIT on the PC Magazine utilities disk.

### BASIC - PRINT:2 TO LPRINT

I was writing a program for a client and came upon another one of those annoying differences between Poly's BASIC.GO and the PC's GWBASIC.EXE. I had a routine that printed loan data to a file for the client, and I nonchalantly said that I could easily give him the option of sending the same report directly to the printer -- (at no extra charge, mind you).

I new that with the Poly, all I had to do was change the file number to 2 and call the subroutine. Here's a sample example in Poly BASIC:

```
100 FILE:4,OPEN,"<3>DATAFILE.TX",OUTPUT
110 F=4 \REM Print the data to the file.
120 GOSUB 200
...
150 FILE:2,LIST
160 F=2 \REM Print the data to the printer
170 GOSUB 200
...
```



```
200 PRINT:F,A$
...
299 RETURN
```

I TRIED to do the same thing in the PC program. I added a statement to open LPT1 for output as # 1 and tried to call the subroutine.

```
100 OPEN "DATAFILE.TXT" FOR OUTPUT AS #1
110 GOSUB 200 \REM Print the data to the file.
...
150 OPEN "LPT1" FOR OUTPUT AS #1
160 GOSUB 200 \REM Print the data to the printer
...
200 PRINT #1,A$
...
299 RETURN
```

It didn't work. It reported a bad file number in the first line that printed the report. But the amazing thing is that it actually printed the whole report. That means that GWBASIC had to finish the entire subroutine, actually using the "bad file number", before it actually reported the error. The whole report was in fact printed!. But the program did "error out" with the bad file number. We can't have the client's program doing that now, can we?

I took the program back to the office and looked up the OPEN, and PRINT statements in the GWBASIC reference manual. Guess what? You can't (legally) open a file channel to the printer device. GWBASIC uses the keyword "LPRINT" for printing to the "line printer". Imagine that -- a whole different keyword to do the same function, but to another device. I vaguely remember that this is standard for most BASIC's. Apple, CP/M's MSBASIC, and a host of other BASIC's all do the same thing

To solve the problem so there would be no error, I had to use my text editor to copy the entire routine that prints to a file, change all the occurrences of "PRINT #1," to "LPRINT ", renumber the copied lines, and, of course, increase the size of the program by a third! It works, but it sure seems inefficient. How come only Poly did it right?

I speculated on whether we could "cheat" by using the ON ERROR routine to trap for the error. If it prints the whole report before the error is registered, that means that it actually processed a lot of lines with the same bad channel. But I haven't gotten around to trying

that trick. Who knows if it could be depended upon. I decide to byte the bullet and copy the routine. If the program were just for me, that would be another story, but for a client? It's gotta be right and work right all the time.

---

### Advertising

Commercial advertising rates are \$50 for a full page, \$25 for a half page, and \$15 for a quarter page. Anything smaller is \$3.00 per column inch. A column is 3-3/4 inches wide by 10 inches tall. A full page is 7-5/8 inches wide. Non-commercial adds by subscribers are free.

---

FOR SALE: 3/4 br home. This house nicely combines the features of a 19th century colonial two-story home with a spacious modern addition. A very large carpeted living-room widely open along part of 2 walls to a full-sized kitchen, a full bath, and a double room that currently serves as a den and the master bedroom make up the downstairs, which is paneled throughout. There is also an enclosed porch, a mud-room, three barns, and an above-ground pool on 3/8th acre of land with nice lilac bush privacy screens from the neighbors. Upstairs are three wall-to-wall carpeted bedrooms and a full bath. The hot water has solar pre-heating, and there is a solar panel on the pool. We're located in the northwest corner of Massachusetts about one mile from Williams College, 5 minutes from Vermont, and 50 minutes from Albany, NY. \$115,000. Call 413-458-3597.

---

Wanted to buy — any and all Poly computers. 88, 8810, 8813, twin, 8824; documentation, software, keyboards, spare parts, etc. — Call Charles Steinhauser — Phone: (404) 739-5081 after 7 pm. EST.

---

FOR SALE: Poly 8810 box with power supply and mother board. \$50 plus shipping. Charles A. Thompson, 2909 Rosedale Avenue, Dallas, Texas 75205-1532, (214)-368-8223.

---

DISKS - MODEMS - PROMS - SOFTWARE - SPELL

1. MAXELL diskettes: 5-1/4" hard sector - \$10 per box.
2. Used diskettes: 5-1/4" 10 hard sector - \$0.50 each.
3. Hayes Micromodem 100 (300 baud S-100 internal modem) - \$20. (If you don't have a modem this is a cheap way to go.)
4. HayesSys modem software (for the Micromodem 100) - \$10.
5. Abstract Systems Exec (Enhancements & bugs corrected) - \$30.
6. Abstract Systems Proms (Enhancements & bugs corrected) - \$35.
7. PolyGlot Library \$6 each volume; 5 or more: \$5 each; ALL: \$99

Abstract Systems, etc., 191 White Oaks Road, Williamstown, MA 01267, Phone: (413) 458-3597

(Send \$1.00 for a complete catalog--(free with any order).)  
(Make check or money order payable to Ralph Kenyon.)

**HELP PROGRAM BACKUP**

HELP file for system program BACKUP.GO

BACKUP.GO copies all files with the "New" bit set.

Syntax:  
BACKUP <d1<Sub1 <d2<Sub2 [\*]

Examples:  
BACKUP 2 3 \*  
BACKUP 4<DATA 3

The optional "\*" means replace the file if it already exists in the destination directory; "d" is the disk drive number. If the "\*" is not on the command line and a file with the same name and extension exists in the destination directory, the program will pause saying:

Output file already exists!  
Should I delete it? (Y or N)

If the answer is "N", then it asks:

OK, give me a new name for the file I'm backing up.  
Filename:

If the answer is "Y", then it replaces the file with the one from the source directory. The replace function is performed as follows: If the files have the same sector length, the source file is written over the destination file on the disk. If the files don't have the same sector length, the destination file is marked deleted and the source file is copied onto the destination disk at the next available disk address. If the destination disk becomes full, the message:

Destination disk is full. Insert new disk,  
then press RETURN to continue.

The copying process continues with the first file in the current directory being processed.

***In This Issue***

Editorial . . . . .	1
Letters . . . . .	1
Printing 8 bits in PM.EXE . . . . .	4
Poly's for sale or donation . . . . .	6
Ralph's Ramblings (more) . . . . .	7
HELP XCOPY . . . . .	8
BASIC - PRINT:2 TO LPRINT . . . . .	8
Advertising . . . . .	9
HELP PROGRAM BACKUP . . . . .	10

**PolyLetter**  
191 White Oaks Road  
Williamstown, MA 01267  
(413) 458-3597

Address Correction Requested

**FIRST CLASS MAIL**



Ralph E. Kenyon, Jr.      EXP:99#9  
Abstract Systems, etc.      184  
191 White Oaks Road  
Williamstown, MA      01267-2256

© Copyright 1992 by Ralph E. Kenyon, Jr.

PolyLetter Editor and Publisher: Ralph Kenyon. Subscriptions: US \$18.00 yr., Canada \$20.00 yr., Overseas \$25.00 yr., payable in US dollars to Ralph Kenyon. Editorial Contributions: Your contributions to this newsletter are always welcome. Articles, suggestions, for articles, or questions you'd like answered are readily accepted. This is your newsletter; please help support it. Non-commercial subscriber adds are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of PolyLetter (1980 through 1991) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 Vol. - \$15, 2 - \$28, 3 - \$40, 4 - \$50, 5 - \$59, 6 - \$67, 7 - \$75; Canada add \$3 shipping, Overseas add \$10. Individual back issues are also available (US: \$3.50, Canada: \$4.00, Overseas: \$5.00).

# PolyLetter



The System-88 Users Newsletter

PolyLetter 92/6

Page 1

NOV/DEC 1992

## Editorial

You'll find some editorializing in the announcement section discussing the Physics programs. I am concerned about the loss of our American advantage in industry and technology. We're so oriented to "the bottom line" and a short term perspective that we don't invest enough in long term development. The Physics demonstration programs being announced might capture the imagination of young children and help them learn math and physics at an early enough age to be more competitive when they get in college and business. For a technologically advanced society, we sure don't put much investment into our children's education, which just happens to be the future.

Keep them cards and letters coming, folks.

## Letters

Dear *PolyLetter*,

November 5, 1992



I'd like any information on ways of keeping a Poly operating. -- Substitute drives - fans - chips - etc. -- Percy Roy, Edmonton, Alberta, Canada.

[Percy, I think it's fair to say that most of the rest of us *PolyLetter* subscribers would like the same thing. We are, perhaps, down to the hard core Poly users by now.

As far as drives are concerned, I believe any PC compatible 360K drive can be used in a Poly. The fact that these drives are double sided won't matter to the Poly because its single density controller ignores the second side, unless you modify the controller itself. The Double density controller was designed for double sided drives anyway, so should not be a

problem. As you know, I have upgraded my system to use 96tpi double sided drives, which I usually pick up at flea-markets or computer shows.

But many places that repair drives on a large scale have these types of drives. My use of floppy drives now is mostly limited to backing up my Hard disk and for archive purposes. But I am slowly transferring data onto PC disks for use with the Poly Emulator. Now that I have a 386DX40, it runs fast enough for me to use it. But I still usually use the Poly itself because the keyboard is more familiar for the macros and screen editor.

Fans are pretty standard and should not be a problem. Chips, on the other hand, may be getting scarce. Probably the best way to keep your Poly running in this regard is to buy up old Poly's as a source of spare parts. But you may be like me. Once you get the spare Poly, it's hard to resist wanting to make it fully functional too. The last issue provided a list of Poly Users who are interested in donating or selling their Polys. You might contact some of them as a source of spare parts.

I have rarely had problems with any of the chips on the Poly boards. My biggest difficulty has been with the Serial I/O mini-card. I've trashed 1488's and 1499's, as well as the buffer chip, which can be replaced with a 74LS367. But those chips are easy to replace. 1488's & 1499's are used in PCs today.

I find the Trenton Computer Festival (TCF) to be a great place to get hardware stuff. It's a 5 hour drive for me each spring. But I usually find the trip worthwhile. I haven't seen much Poly stuff there in the last few years, but last year I remember a memory card, and the year before there was a full Poly system.

The two times I had what could be called a major difficulty, I had a capacitor blow and I

had blown the 12 volt regulator on the CPU card. Both these events actually happened to spare cards I was trying to make functional. The most problem I've had with my actual system itself is its occasional front panel sickness. Usually cleaning the fingers on the board and cleaning or rocking the chips on the cpu and video cards takes care of the problem.

One place that seems to have a lot of parts suitable for the Poly is B. G. Micro, Post Office Box 280298, Dallas, TX, 75228, (214) 271-5546. They have fans, chips, and other hardware parts. They're also the ones that offer a network for up to 3 pc's for \$25.

If you do get one of those used Polys, please let me know so I can clear the name from the file. Let's keep those Polys running. -- Ed.]

Dear Ralph,

November 20, 1992

You could improve *PolyLetter* by using Full Color Glossy Paper. -- Just kidding.

How do you copy a file from the PC hard disk to a floppy that is larger than the floppy capacity without backing up the entire hard disk? -- Michell S. Lippman, Marietta, GA.

[Mitchell, You can use the DOS BACKUP command to backup a single file. It will automat-

```
Microsoft(R) MS-DOS(R) Version 5.00
(C)Copyright Microsoft Corp 1981-1991.

$$HELP BACKUP
Backs up one or more files from one disk to another.

BACKUP source destination-drive: [/S] [/M] [/A] [/F[:size]]
[/D:date[/T:time]] [/L[:drive:][path]logfile]

source           Specifies the file(s), drive, or directory to back up.
destination-drive: Specifies the drive to save backup copies onto.
/S              Backs up contents of subdirectories.
/M             Backs up only files that have changed since the last
              backup.
/A            Adds backup files to an existing backup disk.
/F[:size]      Specifies the size of the disk to be formatted.
/D:date        Backs up only files changed on or after the specified
              date.
/T:time        Backs up only files changed at or after the specified
              time.
/L[:drive:][path]logfile
              Creates a log file and entry to record the backup
              operation.
```

ically break up the file to fit on floppies. You will have to use the RESTORE command to re-load the file if you ever need to. RESTORE will read the diskettes one at a time and reconstruct entire file. -- Ed.]

### Auditing AT&T's Call Anytime Service

I've mentioned before that I used my Poly to make and record long-distance telephone calls. I use the DC-Hayes Micromodem-100 board and custom software I wrote myself as the dialer. I use edit.GO and Gnomus.GO to make re-dialing calls as easy as a few key-strokes. Because of all the long distance calls I made, I subscribed to AT&T's Reach Out America plan.

I also wrote a BASIC program to audit the telephone bill. One reason I did this is so I could keep track of who made what calls, and pro-rate the bill accordingly. I used to pro-rate the base portion of the bill between business and personal calls. The tax people have changed the rules, and one may no longer deduct any portion of the basic service for business calls when there is only one phone. Of course I had to change the program -- it actually simplified it.

I have now converted that system to one using WordPerfect Office Notebook, a Shell macros, and a GW-Basic program for auditing the bill. But last month I switched to a new plan. I now have AT&T's Call Anytime plan. It's cheaper. The way that plan works is one gets 1 hour of calling services for \$10. Additional calls are 20¢ per minute for day calls, and 11¢ per minute for evening and night calls. Of course I had to rewrite my auditing program. The new version of the program is written GW-Basic, but it could easily be converted back to run on the Poly.

Since I first began auditing my bill, I found out that the telephone company gets it right only about 1/3 of the time. They most often get the tax computation wrong. My old program kept finding that they overcharged me by 1¢ in taxes on either the Federal or the state taxes. I wrote several letters about this, but I've basically

been getting stonewalled. I guess it must be beyond the phone company programmer's capability to get it right

I did some rough extrapolations and found out that, on a nation-wide basis, the long distance companies *could* be overcharging their customers to the tune of \$500,000.00 per month! Now that's a nice piece of change. The overcharge appears in the area of the taxes and amounts to 1¢ about every third month -- an amount almost impossible to detect. There are two possibilities regarding how the phone companies are handling this overcharge. They could be keeping it themselves, or they could be giving it to the government. In the first case they are skimming an unreported half Million dollars per month and not paying taxes on it; in the second case we are paying a half million dollars per month in taxes we don't need to pay. Now, it's true that the government needs the money, but I don't think we should pay more taxes than that required by law.

I hear you cry, "How can they get away with such an error?" The mechanism is quite easy actually. I'll make an analogy with sales taxes in department stores. Suppose there is a 3% sales tax on some items in a store. These items cost \$.50 each and you go through the check-out line with two of them. The total sale is \$1.00 and the total tax is 3¢. But suppose you go through the checkout counter twice -- once for each item. Each sale will be \$.50 and the tax on each sale will be 2¢. The total sale will be \$1.00 and the total tax will be 4¢. This is 1¢ more than you were required to pay. This is what the telephone company is doing. They compute the tax on the base portion of the bill; they compute the tax on the sub-total of the calls; and then they add these two taxes together and charge you 4¢ when they should have charged you 3¢. My question was do they pass on the whole 4¢ or do they pass on just 3¢ and pocket the other 1¢? Nobody has given me an answer.

I ran a random sampling program to check out the procedure and found out that a bill divided randomly into two parts and taxed at the 3% rate, should show a 1¢ overcharge an average of one sixth of the time, a 1¢ undercharge an average of one sixth of the time and the correct amount the remaining two thirds of the time. But my experience has been that I have actually been undercharged only twice since I began auditing the bill, and then only

after I began complaining. My actual experience is that the phone company is consistently overcharging me on taxes by 1¢ about once every third month. In a few years there have also been a half a dozen incidents where the charges were off by an unexplainable amount. These errors always showed up on the amount of federal or local tax line. How many people bother to check the computations? When I do find an error, I dutifully call the company and make them credit me with the difference.

I have contacted both the federal excise tax people and the state tax people. Both offices state that the phone company is supposed to compute the tax as a percentage of the total services rendered. Breaking the amount down into two or more individual amounts, computing the tax on each amount, and adding the taxes, is not the correct way to compute the tax portion of the bill. Of course, everyone I talk to says about the same thing "You quibble over a penny?". (An occupational hazard for perfectionists.) But that penny adds up to half a million dollars per month -- six million dollars per year. Who's getting it? The government? The phone companies? Are the phone companies evading taxes on it? That's what would happen if they charged us the wrong way and reported it to the government the correct way.

I heard that one enterprising white-collar criminal robbed a bank by setting up a program to deposit the rounded portion of a cent in a special account. The bank usually keeps these fractions of a cent because they really add up. He eventually got caught and convicted of theft. Should we let the phone companies get away with the same thing?

Oh well, enough pontificating. It's time to get down to the audit program itself.

```

10 DIM D9(1),I9(1),B9(1),Z9(1),X9(1),Y9(1)
20 FOR I = 1 TO 7: B9 = " " + B9 + B9: NEXT
30 INPUT "Year? (89) : ",Y9: IF LEN(Y9) <> 2 THEN 30
40 DIM M9(1),MO9(12): FOR I = 1 TO 12: READ MO9(I): NEXT
50 DATA "Jan","Feb","Mar","Apr","May","Jun"
60 DATA "Jul","Aug","Sep","Oct","Nov","Dec"
70 DIM D9(7): FOR I = 1 TO 7: READ D9(I): NEXT
80 DATA "Saturday","Sunday","Monday"
90 DATA "Tuesday","Wednesday","Thursday","Friday"
100 DIM M9(12): FOR I = 1 TO 12: READ M9(I): NEXT
110 DATA 31,28,31,30,31,30,31,31,30,31,30,31
120 N = 120: REM max number of calls

```

This program allows for dividing the

phone calls up among 5 users. It could easily be changed to do fewer or more.

```

130 U8 = 5: U9 = U8 + 1: REM # of users +1
140 DIM U(N): REM User C$(U(1))
150 DIM C$(U8): FOR I = 1 TO U8: READ C$(I): NEXT
160 DATA "rk","st","bb","pl","as" : REM user codes
170 DIM T(N): REM Type of call T$(T(1))
180 DIM T0$(N): REM Type of call T0$(T(1))
190 DIM T$(N): REM Time
200 DIM D$(N): REM Date
210 DIM N$(N): REM Number
220 DIM D(N): REM Duration
230 DIM A(N): REM Amount
240 M9 = 16: DIM P$(N): REM Place = Who
250 F9 = 0: REM Night fee
260 F8 = 0: REM Evening fee
270 F7 = 10: REM Day fee
280 F6 = 22.08: REM Base fee
290 R0 = 3: REM 3% Federal tax rate
300 R1 = 75: REM 25% AT&T evening rate discount
310 R2 = 90: REM 10% AT&T day rate discount
320 R3 = 5: REM 5% Mass tax rate
330 R4 = 5: REM In state discount
340 R5 = .65: REM AT&T directory assistance
350 R6 = 10: REM AT&T MASS intrastate discount
360 R7 = 95: REM 5% AT&T international discount
370 DATA "DD" : DAY = 1: REM Day calls index
380 DATA "ED" : EVE = DAY+1: REM Evening calls index
390 DATA "ND" : NIT = EVE+1: REM Night calls index
400 DATA "XD" : INTRNAT = NIT+1: REM AT&T INTRNAT calls (Canada)
410 DATA "SD" : STATE = INTRNAT+1: REM AT&T State (617 + 508)
420 DATA "OC" : O1 = STATE+1: REM AT&T Other Charges
430 DATA "DA" : F2 = O1+1: REM Directory (555-1212) calls index
440 DATA "DA" : F3 = F2+1: REM Directory (555-1212) for Mass
445 DATA " " : N1 = F3+1: REM Day time index
450 DATA " " : N4 = N1+1: REM Day charges index
455 DATA " " : N2 = N4+1: REM Evening time index
460 DATA " " : N5 = N2+1: REM Evening charges index
470 DATA " " : N3 = N5+1: REM Night time index
480 DATA " " : N6 = N3+1: REM Night charges index
490 DATA " " : B1 = N6+1: REM AT&T Base charges index
500 DATA " " : D6 = B1+1: REM AT&T state discount
510 DATA " " : D7 = D6+1: REM AT&T MASS intrastate discount
520 DATA " " : AFT = D7+1: REM AT&T Federal Taxes index
530 DATA " " : AFTO = AFT+1: REM AT&T Federal Tax overcharge
540 DATA " " : AMT = AFTO+1: REM AT&T Mass Taxes index
550 DATA " " : AMTO = AMT+1: REM AT&T Mass Tax overcharge
560 DATA " " : A1 = AMTO+1: REM AT&T total index
570 DATA " " : B2 = A1+1: REM NET Base charges index
580 DATA "LD" : S1 = B2+1: REM Local dial (413) calls
590 DATA "LD" : S2 = S1+1: REM CallAround 413 extra
595 DATA "XD" : S3 = S2+1: REM 900 CALLS
600 DATA "OC" : O2 = S3+1: REM NET Other Charges
610 DATA " " : T3 = O2+1: REM NET Federal Taxes index
620 DATA " " : NFTO = T3+1: REM NET Federal Tax overcharge
630 DATA " " : T4 = NFTO+1: REM NET Mass Taxes index
640 DATA "CR" : C0 = T4+1: REM NET credits

```

```

650 DATA "CR" : C3 = C0+1: REM AT&T credits
660 DATA " " : A2 = C3+1: REM NET total index
670 DATA " " : M1 = A2+1: REM MCI calls
680 DATA " " : T5 = M1+1: REM MCI tax
690 DATA " " : M2 = T5+1: REM MCI total
700 DATA "FD" : F1 = M2+1: REM Free dial (800) calls index
710 DATA " " : T9 = F1+1: REM Total index
720 DATA " " : W1 = T9+1: REM Working index
730 DIM T9$(W1): FOR I = 1 TO W1: READ T9$(I): NEXT: REM type
740 DIM N(U9,W1): REM n(i,j) i= rk,st,pl,as
750 INPUT "What month is this?",M9
760 IF M9 = "" OR LEN(M9)<>3 THEN 3520

```

Lines 770-790 conditions the month name to the form Xxx. We used PEEK, POKE, and MEM functions on the Poly, but MEM is not available in PC basics. And I don't know where PEEK and POKE go in PC basics.

```

770 CHAR=ASC(MID$(M9,1,1)):MID$(M9,1,1)=CHR$(CHAR AND 95)
780 CHAR=ASC(MID$(M9,2,1)):MID$(M9,2,1)=CHR$(CHAR OR 32)
790 CHAR=ASC(MID$(M9,3,1)):MID$(M9,3,1)=CHR$(CHAR OR 32)
800 M0 = 0: FOR I = 1 TO 12: IF M9 = M0$(I) THEN M0 = I
810 NEXT: IF M0 = 0 THEN 750
820 REM FILE:2,LIST
      ;FILE:4,OPEN,"5<CALLS<"+Y9+"<"+M9+".TX",INPUT
830 OPEN "C:\BS\"+Y9+"\ "+M9+".TXT" FOR INPUT AS #1:
840 C1 = 0
850 REM FILE:5,OPEN,"5<CALLS<"+Y9+"<"+M9+".TX",INPUT
860 OPEN "C:\BS\"+Y9+"\ "+M9+".DAT" FOR INPUT AS #2
870 OPEN "C:\BS\"+Y9+"\ "+M9+".PRN" FOR OUTPUT AS #3

```

Line 880 begins the main loop which reads and merges the information in files C:\BS\92\DEC.TXT and C:\BS\92\DEC.DAT. File .TXT contains the record of the calls. File .DAT contains the time and price information in the same order. One day I'm going to change it to one file that I just add the time and price info to. The info is read into various arrays defined in lines 170-240.

```

880 IF EOF(1) THEN 1820 ELSE INPUT #1,I19
      : REM IF I19 = "" THEN 1580
890 C1 = C1 + 1: D09 = I19
900 REM File format:
910 REM 91/07/12 21:54 1-802-123-4567 rk Comments
920 REM 1 4 7 10 13 16          31 34
930 T = VAL(MID$(I19,10,2)) * 100 + VAL(MID$(I19,13,2))
940 H9 = DAY: IF T > 1659 THEN H9 = EVE
950 IF T < 800 OR T > 2159 THEN H9 = NIT
960 AMPM9 = "AM": IF T > 1159 THEN T = T - 1200: AMPM9 = "PM"
970 IF T < 100 THEN T = T + 1200
980 T$(C1) = STR$(T)
990 IF LEN(T$(C1)) > 4 THEN T$(C1) = MID$(T$(C1),2,4)
1000 IF LEN(T$(C1)) < 4 THEN T$(C1) = " " + T$(C1): GOTO 1000
1010 T$(C1) = T$(C1) + AMPM9
1020 REM
1030 M0 = VAL(MID$(I19,4,2))
1040 IF M0 = 0 THEN PRINT "Bad month in Month file": GOTO 3520

```

```

1050 REM
1060 D0 = VAL(MID$(119,7,2))
1070 Y = VAL(MID$(119,1,2)) + 1900
1080 M3 = M0: D9 = Y * 365 + D0
1090 L = (Y / 4 = INT(Y / 4) AND Y / 100 <> INT(Y / 100)) OR Y /
400 = INT(Y / 400)
1100 IF M3 > 2 THEN D9 = D9 - L
1110 FOR I = 1 TO 12: IF M3 > 1
      THEN M3 = M3 - 1: D9 = D9 + M9(I)
1120 NEXT: D9=D9+INT((Y-1)/4)-INT((Y-1)/100)+INT((Y-1)/400)
1130 D9 = D9 - INT(D9 / 7) * 7 + 1
1140 IF D9$(D9) = "Saturday" THEN H9 = NIT: GOTO 1330
1150 IF D9$(D9) = "Sunday" AND H9 = DAY THEN H9 = NIT: GOTO 1330

```

Now we've gotta do all kinds of shenanigans to figure out the holidays that are treated as night rate calls during the day hours. We've already taken care of weekends, so the rest only happens if the holiday falls during the week.

```

1160 IF M0 = 1 AND D0 = 1 THEN GOSUB 1800: REM New Years day
1170 REM IF M0 = 5 AND D0 = 3 THEN GOSUB 1800: REM Custom 1991
1180 IF M0 = 7 AND D0 = 4 THEN GOSUB 1800: REM July 4
1190 IF M0 = 11 AND D0 = 11 THEN GOSUB 1800: REM Veterans day
1200 IF M0 = 12 AND D0 = 25 THEN GOSUB 1800: REM Christmas
1210 IF M0 = 12 AND D0 = 31 THEN GOSUB 1800: REM New Years Eve
1220 W0 = INT((D0 + 6) / 7): REM calculate week
1230 REM Now check for Martin Luther King day
1240 IF W0=3 AND D9$(D9)="Monday" AND M0 = 1 THEN GOSUB 1800
1250 REM Now check for Presidents day
1260 IF W0=3 AND D9$(D9)="Monday" AND M0 = 2 THEN GOSUB 1800
1270 REM Now check for Labor day
1280 IF W0=1 AND D9$(D9)="Monday" AND M0 = 9 THEN GOSUB 1800
1290 REM Now check for Thanksgiving
1300 IF W0=4 AND D9$(D9)="Thursday" AND M0 = 11 THEN GOSUB 1800
1310 REM Now check for Columbus Day
1320 IF W0=2 AND D9$(D9)="Monday" AND M0 = 10 THEN GOSUB 1800
1330 IF H9 = DAY THEN T8 = 1 ELSE T8 = 0
1340 REM Strip Year and "- "
1350 N$(C1) = MID$(119,16,14): REM Number
1360 IF MID$(119,16,1) = "/" THEN H9 = S1: T8 = 0: GOTO 1610
      REM Local calls begin with /
1370 IF MID$(119,22,8) = "447-8600" THEN H9 = F1: GOTO 1610
      REM Telephone company office free call
1380 IF MID$(119,22,8) = "555-1611" THEN H9 = F1: GOTO 1610
      REM Telephone company free call
1390 IF MID$(119,18,3) = "800" THEN H9 = F1: GOTO 1610
      REM 800 numbers are free
1400 IF MID$(119,18,12) = "617-555-1212" THEN H9=F3: GOTO 1610
1410 IF MID$(119,18,12) = "508-555-1212" THEN H9=F3: GOTO 1610
      REM instate directory assistance calls
1420 IF MID$(119,18,3) = "///" THEN H9=S1: GOTO 1610: REM 413
      REM instate long distance calls
1422 IF MID$(119,18,3) = "976" THEN H9 = S3
      REM local specially charged calls
1430 IF MID$(119,18,3) = "617" THEN H9 = STATE
1440 IF MID$(119,18,3) = "508" THEN H9 = STATE
      REM instate calls not in long distance plan
1450 IF MID$(119,18,3) = "902" THEN H9 = INTRNAT

```

```

1460 IF MID$(119,18,3) = "506" THEN H9 = INTRNAT
1470 IF MID$(119,18,3) = "719" THEN H9 = INTRNAT
1480 IF MID$(119,18,3) = "418" THEN H9 = INTRNAT
1490 IF MID$(119,18,3) = "514" THEN H9 = INTRNAT
1500 IF MID$(119,18,3) = "819" THEN H9 = INTRNAT
1510 IF MID$(119,18,3) = "613" THEN H9 = INTRNAT
1520 IF MID$(119,18,3) = "416" THEN H9 = INTRNAT
1530 IF MID$(119,18,3) = "519" THEN H9 = INTRNAT
1540 IF MID$(119,18,3) = "705" THEN H9 = INTRNAT
1550 IF MID$(119,18,3) = "807" THEN H9 = INTRNAT
1560 IF MID$(119,18,3) = "204" THEN H9 = INTRNAT
1570 IF MID$(119,18,3) = "306" THEN H9 = INTRNAT
1580 IF MID$(119,18,3) = "403" THEN H9 = INTRNAT
1590 IF MID$(119,18,3) = "604" THEN H9 = INTRNAT
      REM international calls
1600 IF MID$(119,22,8) = "555-1212" THEN H9 = F2
      REM Directory assistance calls $.65 each
1610 T(C1) = H9: T0$(C1) = T9$(H9)
1620 DAY$=STR$(D0): IF LEN(DAY$)>2 THEN DAY$=MID$(DAY$,2)
1630 D$(C1)=M0$(M0)+" "+DAY$
1640 REM Strip Phone #
      REM Now look for the user code
1650 C2 = 0: FOR I=1 TO U8: IF MID$(119,31,2) = C$(I) THEN C2=I
1660 NEXT: IF C2 = 0 THEN PRINT "Bad user in Month file": STOP
1670 U(C1)=C2: P$(C1)=MID$(119,34)+B$: P$(C1)=LEFT$(P$(C1),W9)
1680 IF H9 = F1 THEN D(C1) = 0: A(C1) = 0: GOTO 1740
1690 IF H9 = F2 THEN D(C1) = 0: A(C1) = R5: GOTO 1740
1700 IF H9 = F3 THEN D(C1) = 0: A(C1) = 0: GOTO 1740
1710 REM IF T8=0 THEN 1480
1720 IF H9 = S1 THEN 1740
1730 INPUT #2,D(C1),A(C1): REM get the time and price
      REM now print the line so we can easily find errors
1740 PRINT D$(C1); " "; T$(C1); " "; P$(C1); " "; N$(C1); " ";
1750 PRINT T0$(C1);
1760 PRINT USING "####"; D(C1);
1770 PRINT USING "###.##"; A(C1);
1780 PRINT " "; C$(U(C1))
1790 GOTO 880
1800 IF H9 = DAY THEN H9 = EVE: REM Holiday rates are in effect
1810 RETURN
      REM now look for special additional items
1820 IF EOF(2) THEN 2140 ELSE INPUT #2,119: C1=C1+1: D0$ = 119
1830 IF RIGHT$(D0$,1)="0" THEN T(C1)=S1:T0$(C1)=T9$(INTRNAT)
      :GOTO 2010
1840 IF RIGHT$(D0$,1) = "W" THEN T(C1) = 02: T0$(C1) = T9$(02)
      : GOTO 2010 :REM New England Telephone additional charge
1850 IF RIGHT$(D0$,1) = "A" THEN T(C1) = 01: T0$(C1) = T9$(01)
      : GOTO 2010 :REM AT&T additional charge
1860 IF RIGHT$(D0$,1) = "C" THEN T(C1) = C3: T0$(C1) = T9$(C3)
      : GOTO 2010 :REM AT&T credit
1870 IF RIGHT$(D0$,1) = "c" THEN T(C1) = C0: T0$(C1) = T9$(C0)
      : GOTO 2010 :REM New England Telephone credit
1880 IF RIGHT$(D0$,1) = "4" THEN T(C1) = S2: T0$(C1) = T9$(S2)
      : GOTO 2010 :REM Call around 413 charge
1890 IF RIGHT$(D0$,1)="m" THEN T(C1) = M1: T0$(C1) = T9$(M1)
      REM MCI separate call
1900 IF RIGHT$(D0$,1)="n" THEN T(C1) = S1: T0$(C1) = T9$(S1)
1910 IF RIGHT$(D0$,1)="a"

```

```

      THEN T(C1) = INTRNAT: T0$(C1) = T9$(INTRNAT)
2120 IF T0$(C1) = ""
      THEN PRINT "Bad code in month file": GOTO 3520
2130 D$(C1) = LEFT$(D0$,6): D0$=MID$(D0$,8,127): REM Date
2140 T$(C1) = LEFT$(D0$,6): D0$=MID$(D0$,8,127): REM Time
2150 P$(C1) = LEFT$(D0$,9): D0$=MID$(D0$,9+2,127): REM Who
2160 N$(C1) = LEFT$(D0$,12): D0$ = MID$(D0$,14,127): REM Number
2170 T0$(C1) = LEFT$(D0$,2): REM Code
2180 D0$ = MID$(D0$,4,127): REM Duration and amount
2190 D(C1) = VAL(LEFT$(D0$,3))
2200 D0$ = MID$(D0$,5,127)
2210 A(C1) = VAL(LEFT$(D0$,6))
2220 D0$ = RIGHT$(D0$,3)
2230 C2 = 0: FOR I = 1 TO UB
2240 IF LEFT$(D0$,2) = C$(I) THEN C2 = 1
2250 NEXT
2260 IF C2=0
      THEN PRINT "Bad user in month file": STOP: GOTO 3520
2270 U(C1) = C2
2280 PRINT D$(C1): " "; T$(C1): " "; P$(C1): " "; N$(C1): " ";
2290 PRINT T0$(C1);
2300 PRINT USING "####"; D(C1);
2310 PRINT USING "###.##"; A(C1);
2320 PRINT " "; C$(U(C1))
2330 GOTO 1820
      REM Now we print out the report - first the detail calls
2340 PRINT "There are"; C1; " lines to print."
2350 FOR I=1 TO U9: FOR J=1 TO W1: N(I,J) = 0: NEXT: NEXT: I=1
2360 IF I > C1 THEN 2290
2370 PRINT #3,D$(I): " "; T$(I): " "; P$(I): " "; N$(I): " ";
2380 PRINT #3,T0$(I);
2390 PRINT #3,USING "####"; D(I);
2400 IF T(I) = STATE THEN PRINT #3,"t": : GOTO 2230
2410 IF T(I) = INTRNAT THEN PRINT #3,"+": : GOTO 2230
2420 IF T(I) < INTRNAT THEN PRINT #3,"T": ELSE PRINT #3," ";
2430 PRINT #3,USING "###.##"; A(I);
2440 PRINT #3," "; C$(U(I))
2450 REM add summing routines here
2460 IF T(I)=DAY THEN N(U(I),N1)=N(U(I),N1)+D(I): REM add time
2470 IF T(I)=EVE THEN N(U(I),N2)=N(U(I),N2)+D(I): REM add time
2480 IF T(I)=NIT THEN N(U(I),N3)=N(U(I),N3)+D(I): REM add time
2490 N(U(I),T(I)) = N(U(I),T(I)) + A(I): REM add $ amount
2500 I = I + 1: GOTO 2160
2510 FOR J = 1 TO UB: FOR I = 1 TO T9
2520 N(U9,I) = N(U9,I) + N(J,I)
2530 NEXT: NEXT: REM compute total column
2540 PRINT #3,CHR$(12)
2550 PRINT #3,CHR$(13);TAB(43);M$;"-";Y$;" Summary";CHR$(13)
2560 PRINT #3,TAB(32);
2570 FOR I=1 TO UB: PRINT #3," ";C$(I);
      : NEXT: PRINT #3," Total"
2580 N(1,B1)=F9+F8+F7: N(U9,B1) = N(1,B1): REM Base charges
2590 X$ = "AT&T Monthly Charges": K = B1: GOSUB 3490
2600 IF N(U9,O1)<>0
      THEN X$ = "AT&T Other Charges": K = O1: GOSUB 3490
2610 IF N(U9,C3)<>0
      THEN X$ = "AT&T Other Credits": K = C3: GOSUB 3490
2620 FOR I=1 TO U9: N(1,W1) = N(1,STATE) + N(1,INTRNAT): NEXT
2630 X$ = "AT&T Itemized Calls": K = W1: GOSUB 3490
2640 FOR I = 1 TO U9: N(1,T9)=N(1,N1)+N(1,N2)+N(1,N3) : NEXT
2650 X$ = "Time used      ": K = T9 :GOSUB 3490
2660 ALLOT=60
2670 X$="Day Calls Tolls": K=DAY:GOSUB 3490: REM Before discount
2680 X$="Day Calls Time ": K = N1 :GOSUB 3490
2690 N(U9,N4)=N(U9,N1)-ALLOT : REM ADAY=DAY-ALLOT
2700 REM IF ADAY<0 THEN ALLOT=-ADAY :ADAY=0
2710 IF N(U9,N4)<0 THEN ALLOT = - N(U9,N4)
      : FOR I=1 TO U9: N(1,N4)=0: NEXT :GOTO 2466
2720 REM DAY CALLS TIME EXCEEDS ALLOTMENT,
      SO MUST PROPORTION IT OUT
2730 FOR I=1 TO U9: N(1,W1)=ALLOT*N(1,N1)/N(U9,N1)
      : NEXT :ALLOT=0
2740 FOR I = 1 TO U9: N(1,N4) = (N(1,N1)-N(1,W1))*2 :NEXT
2750 X$ = "ADDTL DAY CALLS ": K = N4: GOSUB 3490: REM Day
2760 FOR I = 1 TO U9: N(1,W1) = (N(1,DAY)-N(1,N4)) :NEXT
2770 X$="Day calls savings ": K=W1: GOSUB 3490: REM Day
2780 X$="Evening Calls Tolls": K=EVE: GOSUB 3490
      : REM Before discount
2790 X$ = "Evening Calls Time ": K = N2 :GOSUB 3490
2800 N(U9,N5)=N(U9,N2)-ALLOT : REM AVEE=EVE-ALLOT
2810 REM IF AVEE<0 THEN ALLOT=-AVEE :AVEE=0
2820 IF N(U9,N5)<0 THEN ALLOT = - N(U9,N5)
      : FOR I=1 TO U9: N(1,N5) = 0: NEXT : GOTO 2602
2830 REM EVE CALLS TIME EXCEEDS ALLOTMENT,
      SO MUST PROPORTION IT OUT
2840 FOR I=1 TO U9: N(1,W1) = ALLOT*N(1,N2)/N(U9,N2)
      :NEXT : ALLOT=0
2850 FOR I = 1 TO U9: N(1,N5) = (N(1,N2)-N(1,W1))*11 :NEXT
2860 X$ = "ADDTL EVENING CALLS ": K = N5: GOSUB 3490: REM Eve
2870 FOR I = 1 TO U9: N(1,W1) = (N(1,EVE)-N(1,N5)) :NEXT
2880 X$ = "Evening calls savings ": K = W1: GOSUB 3490: REM eve
2890 X$ = "N/Weekend Calls Tolls": K = NIT
      : GOSUB 3490: REM Before discount
2900 X$ = "N/Weekend Calls Time ": K = N3 :GOSUB 3490
2910 N(U9,N6)=N(U9,N3)-ALLOT : REM ANIT=NIT-ALLOT
2920 REM IF ANIT<0 THEN ALLOT=-ANIT :ANIT=0
2930 IF N(U9,N6)<0 THEN ALLOT = - N(U9,N6)
      : FOR I=1 TO U9: N(1,N6)=0: NEXT :GOTO 2624
2940 REM NIT CALLS TIME EXCEEDS ALLOTMENT,
      SO MUST PROPORTION IT OUT
2950 FOR I=1 TO U9: N(1,W1)=ALLOT*N(1,N3)/N(U9,N3)
      : NEXT : ALLOT=0
2960 FOR I=1 TO U9: N(1,N6) = (N(1,N3)-N(1,W1))*11 :NEXT
2970 X$ = "ADDTL N/WEEKEND CALLS ": K = N6: GOSUB 3490: REM Eve
2980 FOR I=1 TO U9: N(1,W1) = (N(1,NIT)-N(1,N6)) :NEXT
2990 X$ = "N/Weekend calls savings ": K=W1: GOSUB 3490: REM eve
3000 FOR I=1 TO U9: N(1,D6)=INT(R4 * N(1,STATE)+.5)/100: NEXT
3010 IF N(U9,D6)<>0
      THEN X$="AT&T In-state credit": K=D6: GOSUB 3490
3020 FOR I = 1 TO U9: N(1,A1) = N(1,N4): REM Day
3030 N(1,A1) = N(1,A1) + N(1,N5): REM Evening
3040 N(1,A1) = N(1,A1) + N(1,N6): REM Night charges
3050 N(1,A1) = N(1,A1) - N(1,D6): REM instate credit
3060 NEXT: X$ = "Anyhour Plan": K = A1: GOSUB 3490
3070 FOR I=1 TO U9: N(1,W1)=N(1,DAY)+N(1,EVE)+N(1,NIT)-N(1,A1)
3080 NEXT

```



```

2700 X$ = "Anyhour Savings": K = W1: GOSUB 3490
2710 IF N(U9,INTRNAT) <> 0
    THEN X$ = "International calls": K = INTRNAT: GOSUB 3490
2720 FOR I=1 TO U9
    : N(I,INTRNAT)=INT(R7*N(I,INTRNAT)+.5)/100: NEXT
2730 IF N(U9,INTRNAT)<>0
    THEN X$="Discounted International calls"
    : K=INTRNAT: GOSUB 3490
2760 IF N(U9,F2)<>0 THEN X$="Directory Assistance"
    : K=F2: GOSUB 3490
2770 FOR I=1 TO U9: N(I,D7)=INT(R6*N(I,STATE)+.5)/100: NEXT
2780 IF N(U9,D7)<>0
    THEN X$="Intrastate discount ": K=D7: GOSUB 3490
2790 FOR I = 1 TO U9: N(I,A1) = N(I,A1) + N(I,INTRNAT)
2800 N(I,A1) = N(I,A1) + N(I,STATE): REM State
2820 N(I,A1) = N(I,A1) - N(I,D7): REM Discount
2830 N(I,A1) = N(I,A1) + N(I,F2): NEXT: REM Information
2840 X$="AT&T Calling Services": K=A1: GOSUB 3490: T6=N(U9,A1)
2850 FOR I=1 TO U9: N(I,A1)=N(I,A1)+N(I,B1): NEXT: REM Base
2860 FOR I=1 TO U9: N(I,AFT)=INT(R0*N(I,A1)+.5)/100: NEXT
2870 X$ = "AT&T Federal Taxes": K = AFT: GOSUB 3490
2880 Z=INT(R0*T6+.5)/100+INT(R0*(F9+F8+F7)+.5)/100-N(U9,AFT)
2890 IF ABS(Z) < .005 THEN 2940
2900 N(U9,AFTO) = ABS(INT(100*Z+.5)/100)
2910 X$ = "* AT&T Federal Tax overcharge *"
2920 IF Z<0 THEN X$ = "* AT&T Federal Tax undercharge *"
2930 K = AFTO: GOSUB 3490: T6 = N(U9,A1)
2940 FOR I=1 TO U9: N(I,AMT) = INT(R3*N(I,A1) + .5) / 100: NEXT
2950 X$ = "AT&T State Taxes": K = AMT: GOSUB 3490
2960 Z=INT(R3*T6+.5)/100+INT(R3*(F9+F8+F7)+.5)/100-N(U9,AMT)
2970 IF ABS(Z) < .005 THEN 3020
2980 N(U9,AMTO) = ABS(INT(100*Z+.5)/100)
2990 X$ = "* AT&T State Tax overcharge *"
3000 IF Z<0 THEN X$ = "* AT&T State Tax undercharge *"
3010 K = AMTO: GOSUB 3490
3020 FOR I=1 TO U9: N(I,A1) = N(I,A1) + N(I,AFT) + N(I,AMT)
3030 N(I,A1) = N(I,A1) + N(I,O1): REM Add other charges
3040 N(I,A1) = N(I,A1) - N(I,C3): NEXT: REM subtract credits
3050 X$ = "AT&T Total Charges": K = A1: GOSUB 3490
3060 PRINT #3,""
3070 N(I,B2) = F6: N(U9,B2) = N(I,B2): REM NET Base charges
3080 X$ = "NET Monthly Charges": K = B2: GOSUB 3490
3090 IF N(U9,O2)<>0
    THEN X$="NET Other Charges": K=O2: GOSUB 3490
3100 IF N(U9,C0)<>0
    THEN X$="NET Other Credits": K=C0: GOSUB 3490
3110 X$ = "CallAround 413 extra": K = S2: GOSUB 3490
3120 FOR I=1 TO U9: N(I,A2)=N(I,S1)+N(I,S2)+N(I,S3): NEXT
3130 X$ = "NET Calling Services": K = A2: GOSUB 3490
3140 FOR I=1 TO U9: N(I,T4) = INT(R3*N(I,A2)+.5)/100: NEXT
3150 FOR I=1 TO U9: N(I,A2) = N(I,A2) + N(I,B2): NEXT
3155 FOR I=1 TO U9: N(I,W1) = N(I,S1) + N(I,S2) + N(I,B2): NEXT
3160 FOR I=1 TO U9: N(I,T3) = INT(R0*N(I,W1)+.5)/100 : NEXT
3170 X$ = "NET Federal Taxes": K = T3: GOSUB 3490
3180 Z=INT(R0*(N(U9,A2)-F6)+.5)/100+INT(R0*F6+.5)/100-N(U9,T3)
3190 IF ABS(Z) < .005 THEN 3230
3200 N(U9,WFTO)=ABS(INT(100*Z+.5)/100)
    :X$="* NET Federal Tax overcharge *"
3210 IF Z<0 THEN X$ = "* NET Federal Tax undercharge *"
3220 K = WFTO: GOSUB 3490
3230 X$ = "NET State Taxes": K = T4: GOSUB 3490
3240 FOR I = 1 TO U9
3250 N(I,A2) = N(I,A2) + N(I,T3) + N(I,T4): REM add taxes
3260 N(I,A2) = N(I,A2) + N(I,O2): REM Add other
3270 N(I,A2) = N(I,A2) - N(I,C0): NEXT: REM subtract credits
3280 X$ = "NET Total Charges": K = A2: GOSUB 3490
3290 IF N(U9,M1) = 0 THEN 3370
3300 PRINT #3,""
3310 X$ = "Other carrier calls": K = M1: GOSUB 3490
3320 FOR I = 1 TO U9
3330 N(I,T5) = INT(R0 * N(I,M1) + .5) / 100: NEXT
3340 X$ = "Other carrier taxes": K = T5: GOSUB 3490
3350 FOR I = 1 TO U9: N(I,M2) = N(I,M1) + N(I,T5): NEXT
3360 X$ = "Other carrier total": K = M2: GOSUB 3490
3370 FOR I = 1 TO U9: N(I,T9) = N(I,A1): REM AT&T Charges
3380 N(I,T9) = N(I,T9) + N(I,A2): REM NET Charges
3390 N(I,T9) = N(I,T9) + N(I,M2): REM MC1 Charges
3400 NEXT
3410 PRINT #3,""
3420 X$ = "Total": K = T9: GOSUB 3490: REM Grand total
3430 PRINT #3,""
3440 PRINT #3,CHR$(13); "Checks: ";
3450 PRINT #3,USING "####.##"; N(U9,T9) - N(3,T9) - N(4,T9);
3460 PRINT #3,USING "####.##"; N(3,T9) + N(4,T9);
3470 PRINT #3,"."; CHR$(12)
3480 RESET: PRINT "Back to DOS.": : SYSTEM
3490 PRINT #3,X$; TAB(32);
3500 FOR J = 1 TO U9: PRINT #3,USING "####.##"; N(J,K); : NEXT:
PRINT #3,""
3510 RETURN
3520 PRINT I$
3530 PRINT "Telephone Accounting month file format."
3540 PRINT "0000000001111111111
22222222223333333333444444444455555555566"
3550 PRINT "1234567890123456789
012345678901234567890123456789012345678901"
3560 PRINT "Mmm DD TTTTtt 12345
67890123456 AAA-XXX-NNNN CC NNN NN.NN cca"
3570 PRINT "Mmm DD TTTTtt 12345
67890123456 AAA-XXX-NNNN CC NNN NN.NN ccn"
3580 PRINT "Mmm DD TTTTtt 12345
67890123456 AAA-XXX-NNNN CC NNN NN.NN ccm"
3590 PRINT "DEC 4 912PM Steph
anie at work 901-423-1339 ED 12 1.22 rka"
3600 PRINT "NN.NN cco (AT&T Operator call charges)"
3610 PRINT "NN.NN ccA (AT&T additional charges)"
3620 PRINT "NN.NN ccN (NET additional charges)"
3630 PRINT "NN.NN cc4 (CallAround 413 charges)"
3640 PRINT "NN.NN ccc (NET credits)"
3650 STOP: PRINT "Back to DOS": RESET: SYSTEM

```

---

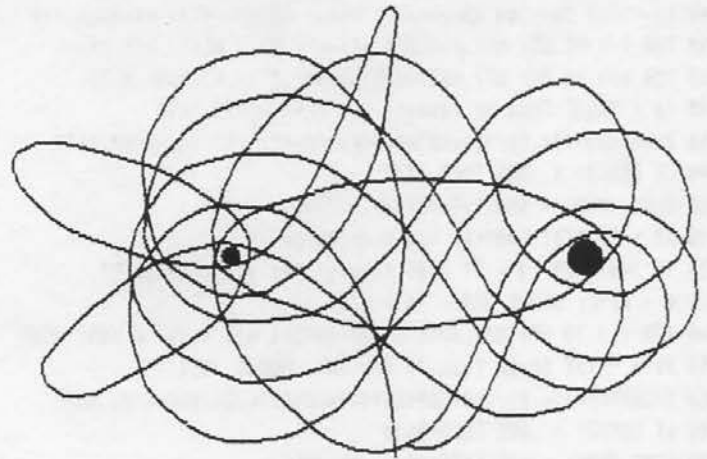
**Annual Index - 1992**

16K DRAM to 64K DRAM Conversion	Ralph Kenyon	92/2/06
486 Upgrade for 286 - Announcement	Evergreen Tech.	92/3/03

ANIMATION - Announcement	FracTerm Inc.	92/4/08
APC Software Adds Postal Bar Codes	Electronic Tech.	92/2/06
Auditing AT&T's Call Anytime Service	Ralph Kenyon	92/6/02
BASIC - PRINT:2 TO LPRINT	Ralph Kenyon	92/5/08
BugNote: 43.0 PM Printer	Abstract Systems	92/4/09
by Leap(Year)s and Bounds	Ralph Kenyon	92/1/01
Changing Sio.PS to print 8 bits	Ralph Kenyon	92/5/02
Chaos Physics Academic Software	announcement	92/6/08
Communications - Astronomy BB	Jim Brennon	92/1/01
Comparing Fractal Images to Other File Formats		92/4/08
Computing your Mortgage	Ralph Kenyon	92/1/07
Dedicated Employees (Swamp)	Anonymous	92/4/05
Dimensioning Variables	Ralph Kenyon	92/2/02
Displaying Fractally Compressed Images - Announcement		92/4/07
Editorial - Call or Write PolyLetter	Ralph Kenyon	92/4/01
Editorial - Change to Masthead	Ralph Kenyon	92/5/01
Editorial - developments in PCs	Ralph Kenyon	92/3/01
Editorial Education and Physics	Ralph Kenyon	92/6/01
Editorial - Input request	Ralph Kenyon	92/1/01
Editorial - Transition to PC	Ralph Kenyon	92/2/01
Estimating Mortgages	Ralph Kenyon	92/3/02
Fractal Compression - Announcement	FracTerm Inc.	92/4/07
Groundhog Day	Bit Bucket (rk)	92/1/10
HELP COMMAND GET	Abstract Systems	92/4/08
HELP COMMAND SAVE	Abstract Systems	92/4/09
HELP for DOS BACKUP command	Microsoft	92/6/02
HELP PROGRAM BACKUP	Abstract Systems	92/5/10
HELP XCOPY	Microsoft	92/5/08
Hypertext catalog	Best Power Tech.	92/6/09
Law and murder-mystery game OBJECTION!	announcement	92/6/09
Letter - Switching to PCs	Ron Moffatt	92/2/01
Letters - How to use DOS BACKUP?	Michell S. Lippman	92/6/01
Letters Keeping Poly Running	Earl Gilbreath	92/4/01
Letters - Keeping the Poly running	Percy Roy	92/6/01
Letters - Laser printer	Jim Salinger	92/5/01
Letters - Testing PM on a 486	Bob Bybee	92/3/01
Managing Your Money	Andrew Tobias	92/6/09
Mister Squock Portrait (large)	DrawPerfect	92/5/05
PM Printer Bug Testing	Abstract Systems	92/4/09
Poly Meta (Part 1)	Ralph Kenyon	92/2/08
Poly Meta (Part 2)	Ralph Kenyon	92/3/04
Poly Trojans and Viruses	Bit Bucket (rk)	92/3/10
Poly's for sale or donation	PolyLetter	92/5/06
Polymorph Virus	Bit Bucket (rk)	92/3/10
Postal Barcodes (part 2)	Ralph Kenyon	92/1/03
Postal Barcodes (part 3)	Ralph Kenyon	92/2/02
Preferred Univocal Notation	Bit Bucket (rk)	92/2/10
Printing 8 bits in PM.EXE	Ralph Kenyon	92/5/04
Ralph's Ramblings (more)	Ralph Kenyon	92/5/07
Ralph's Ramblings	Ralph Kenyon	92/4/01
REAL-TIME GRAPHICS - Announcement	FracTerm Inc.	92/4/07
Turning Poly's ECHO off (and on)	Ralph Kenyon	92/4/03
Two printers for PM	Ralph Kenyon	92/4/02
Virus Authorship Made Easy	I. C. S. A	92/3/09
Windows wins?	Bit Bucket	92/6/10
WordPerfect Works - Announcement	WordPerfect Corp.	92/3/02

## Announcements

Physics Academic Software, American Institute of Physics, 335 East 45th Street, New York, NY 10017-3483, now offers a number of software packages for use in demonstrating and teaching physics. *PolyLetter* has received a number of packages, which I shall just list briefly here. The new science of Chaos has been in the news during the last few years, and Physics Academic Software has sent *PolyLetter* a review copy of software which demonstrates Chaos effects. It's interesting to watch a planet orbit around two suns. Watching it shows clearly why we will be very unlikely to ever find planets in binary systems. Here's what the path looked like for a while.



As you can see, the planet gets too near both suns. The simulation eventually has the planet crashing into one sun. If we extrapolate this situation to the formation of a binary star system, we would expect the gasses around the stars to travel in similar paths and eventually get absorbed by the two suns. No planet would form. The orbit shows what would happen to a captured planet. So, when your travel agent suggests a visit to a quiet planet with twin suns, you know you're being conned. (Pay attention, Charles.)

The Chaos Demonstration software package demonstrates 17 other different effects. Here's a list of the demonstrations included.

- A: Driven Pendulum
- B: Nonlinear Oscillator
- C: Van der Pol Equation
- D: Three-Body Problem
- E: Magnetic Quadrupole
- F: Lorentz Attractor
- G: Logistic Map

H: Predator-Prey  
 I: Chirikov Map  
 J: Henon Map  
 K: Mandelbrot Set  
 L: Julia Sets  
 M: Diffusion  
 N: Noise  
 O: Deterministic Fractals  
 P: Random Fractals  
 Q: Weierstrass Function  
 R: Game of Life

The orbit of a planet about 2 suns is an example of the "three body problem". As you can see, the orbit is total chaos. (I couldn't resist that.) Here's the list of program packages offered. Maybe things like this can get our kids (and grandchildren) interested in Physics early enough to make a difference.

Mathplot . . . . .	\$49.95
Fit Kit (for fitting polynomials to data)	\$49.95
Gradebook . . . . .	\$49.95
Physics Simulation Programs . . . . .	\$49.95
Physics Demonstrations . . . . .	\$49.95
Thermodynamics Lecture Demonstrations	\$49.95
Orbits . . . . .	\$49.95
Spacetime (special relativity) . . . . .	\$59.95
Maxwell (electromagnetic theory) . . . . .	\$59.95
Vibrational Modes . . . . .	\$24.95
Atomic Scattering . . . . .	\$24.95
Quantum Scattering . . . . .	\$24.95
Relativistic Collision . . . . .	\$59.95
Chaos Demonstrations 2.0 . . . . .	\$69.95
Chaos Simulations . . . . .	\$69.95
Chaotic Dynamics Workbench . . . . .	\$69.95
Chaos Data Analyzer . . . . .	\$99.95

TransMedia, Inc., 27404 Drake Road, Farmington Hills, MI 48331, (313) 553-9100, has announced that a computer game they publish has received professional certification -- the Law and murder-mystery game **OBJECTION!** It is the same courtroom simulation used to train trial lawyers. It has now been certified by 10 state bars for Continuing Legal Education. Three versions are available. Non-lawyers pay \$49.95, and lawyers pay \$99.00. The CLE version costs \$249.00. Available for PC and Macintosh. Call 1-800-832-4980 for more info.

MECA Software, Inc., (Andrew Tobias), 55 Walls Drive, Box 910, Fairfield, CT 06430-1912, announced version 9.0 of their *Managing Your Money* Software package. The package can handle different users with their own files and passwords. It includes a calendar, card file, calculator, checkbook and money management

accounts, tax accounting, insurance and estate planning, financial analysis, stock portfolio management, and net-worth calculations. It can present its data using graphs and charts. It also supports check writing, *CheckFree* electronic bill paying, and importing data from on line services such as PRODIGY. What's more it's on sale for \$29.95 (plus a non-refundable \$8.50 shipping) with a money back guarantee until 2/28/93. Call 1-800-945-3023 for more info.

Best Power Technology, Inc., Post Office Box 280, Necedah, WI 45646, announced a new hypertext catalog on disk. The catalog requires a DOS compatible PC with a VGA display; it supports a mouse. The catalog is available free of charge. If you need an uninterruptible power supply (UPS) so you don't lose the data your Poly is working on when the power goes out, this company may have what you need. But I think the hypertext catalog on disk is the way many companies will go in the future. Call 1-800-356-5794 for more info or to request a free catalog on disk.

---

### Advertising

Commercial advertising rates are \$50 for a full page, \$25 for a half page, and \$15 for a quarter page. Anything smaller is \$3.00 per column inch. A column is 3-3/4 inches wide by 10 inches tall. A full page is 7-5/8 inches wide. Non-commercial ads by subscribers are free.

---

Wanted to buy — any and all Poly computers. 88, 8810, 8813, twin, 8824; documentation, software, keyboards, spare parts, etc. — Call Charles Steinhauser - Phone: (404) 739-5081 after 7 pm. EST.

---

FOR SALE: Poly 8810 box with power supply and mother board. \$50 plus shipping. Charles A. Thompson, 2909 Rosedale Avenue, Dallas, Texas 75205-1532, (214)-368-8223.

---

DISKS - MODEMS - PROMS - SOFTWARE - SPELL

1. MAXELL diskettes: 5-1/4" hard sector - \$10 per box.
2. Used diskettes: 5-1/4" 10 hard sector - \$0.50 each.
3. Hayes Micromodem 100 (300 baud S-100 internal modem) - \$20.  
(If you don't have a modem this is a cheap way to go.)
4. HayesSys modem software (for the Micromodem 100) - \$10.
5. Abstract Systems Exec (Enhancements & bugs corrected) - \$30.
6. Abstract Systems Proms (Enhancements & bugs corrected) - \$35.
7. PolyGlot Library \$6 each volume; 5 or more: \$5 each; ALL: \$99

Abstract Systems, etc., 191 White Oaks Road,  
 Williamstown, MA 01267, Phone: (413) 458-3597

(Send \$1.00 for a complete catalog--(free with any order).)  
(Make check or money order payable to Ralph Kenyon.)

**Bit Bucket**

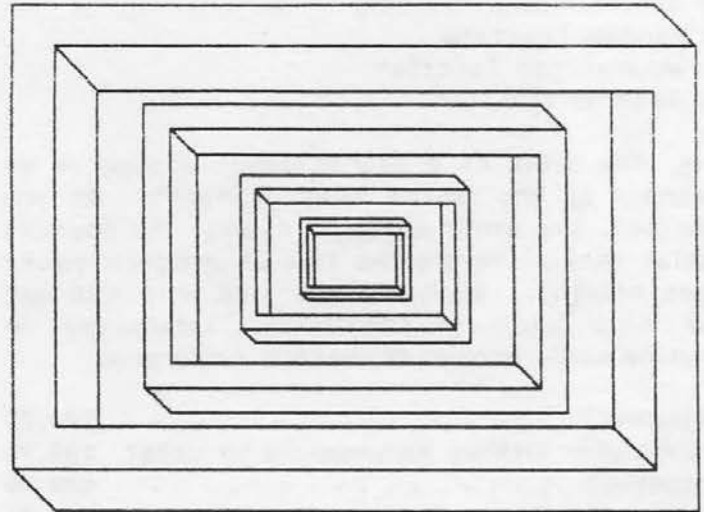
Here's a tidbit from Charles Steinhauser. Charles's Prediction: "According to the Prophets, Windows NT is the latest and greatest. Allegedly it is the operating system of the future."

Interestingly enough, Al Levy, editor of *The Stack* -- the newsletter for the Long Island Computer Association (LICA) agrees with Charles. Al said, "The war is over. Windows is the winner. New upgrades to many major products are only for the Windows version."

**In This Issue**

Editorial . . . . .	1
Letters . . . . .	1
DOS BACKUP . . . . .	2
Auditing AT&T's Call Anytime Service . . . . .	2
Annual Index - 1992 . . . . .	7
Announcements . . . . .	8
Physics Academic Software . . . . .	8

Law and murder-mystery game	
<b>OBJECTION!</b> . . . . .	9
Managing Your Money . . . . .	9
Best Power Technology . . . . .	9
Advertising . . . . .	9
Bit Bucket . . . . .	10



What's happening here?

**PolyLetter**  
191 White Oaks Road  
Williamstown, MA 01267  
(413) 458-3597

Address Correction Requested

**FIRST CLASS MAIL**



Ralph E. Kenyon, Jr.	EXP:99#9
Abstract Systems, etc.	184
191 White Oaks Road	
Williamstown, MA	01267-2256

© Copyright 1992 by Ralph E. Kenyon, Jr.

PolyLetter Editor and Publisher: Ralph Kenyon. Subscriptions: US \$18.00 yr., Canada \$20.00 yr., Overseas \$25.00 yr., payable in US dollars to Ralph Kenyon. Editorial Contributions: Your contributions to this newsletter are always welcome. Articles, suggestions, for articles, or questions you'd like answered are readily accepted. This is your newsletter; please help support it. Non-commercial subscriber adds are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* (1980 through 1991) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 Vol. - \$15, 2 - \$28, 3 - \$40, 4 - \$50, 5 - \$59, 6 - \$67, 7 - \$75; Canada add \$3 shipping, Overseas add \$10. Individual back issues are also available (US: \$3.50, Canada: \$4.00, Overseas: \$5.00).