PolyLetter

Editorial

Why should I review IBM type software in PolyLetter? Many of our readers have both Poly and IBM compatible machines, and, like it or not, we will eventually have to replace our Poly's. (Poly's closing this January is likely to hasten the process because repair service will be less available.) I have now accumulated no less than ten Poly's and expect to be able to keep at least one running for a long time yet. Don't fear, folks, Poly may be closing, but PolyLetter ISN'T.

However, to help ease the transition for those of us who can't hold out forever, I will be on the lookout for IBM compatible software which provides the kind of quality we have come to expect of Poly software. If any of you find IBM compatible software which helps ease the transition, tell us about it so we can share it with other Poly users.

Most effective Poly software was user written by the pioneers in the field who were 'the first kids on the block with one'. We had to work to make the Poly work for us. But, when we did, it worked for us grandly.

The new breed of computer user buys software rather than writes it. In a sense, we could say we paid our dues with our early Poly software so we are now entitled to get our IBM compatible software the easy way (buy it).

However, we do have high standards as a result of where we came from. We expect good software to be economical in its use of both memory and disk-space. We expect it to be "clean" by not putting anything on the screen that isn't related to the immediate input/output required by the user.

Most of all, we are accustomed to having software which does exactly what we want it to do (after all, most of us wrote our own to our own exact specifications).

So, let us share with each other information on any IBM compatible software which seems to meet these high Poly standards. PolyLetter will also attempt to obtain any such suitable software at discount prices.

LETTERS

Dear Ralph: January 27, 1989

I appreciate the information you offered during our recent 7:45 AM telephone conversation.

I have an extra MS 8" 2-drive unit on another POLY. Under version 96, is there much involved with setting up both MS (affording 4 drives) to run off the same 8813 ???

Finally, another testimonial to the importance and immeasureable benefits from PolyLetter. Your recent series on UNSAVEP and Bob's concerning machine code were enlightening, educational, practical and timely. My breadth of understanding has octupled and opportunities increased hexi-fold as the result of reading (studying) PolyLetter. I hope both me and my equipment can hold out long enough until I can transfer all my "deadline" applications to an IBM clone and have the POLY for strictly developmental and educational pursuits.

How about publishing a current listing of subscribers with their addresses and phone numbers? --- J. Earl Gilbreath, Jr., Savannah, GA.

[The 88-MS Users Manual is very sketchy, but the 8 inch controller has three address lines going to the drive and the drives have 4 drive select lines. It seems to me that the two MS units can just be daisy chained together, and the drives addressed to 3 and 4 (2 and 3 if the drive address lines are numbererd 0-3) on the second MS unit. All you would have to do is add a connector to the existing cable to the MS, or make up a longer cable with two, sufficiently spaced, connectors in the MS end of the new cable. As for the addresses, here they are. -- Ed.]

Poly Users

Dr. Michael Aquino 8902
Post Office Box 4507
Saint Louis, MO 63108

Arnell K. Baker 8905
4843 Hernandez Drive
Guadalupe, CA 93434
805-343-1897

Joanne Bransdon 9900
FOGHORN, FOG Users Group
Post Office Box 3474
Daly City, CA 94015-0474
415-755-2000

Mike Breza 8905
109 Maplelawn, S.W.
Wyoming, MI 49508
616-534-5573

Dr. L. G. Briarty 8905
Department of Botany
Nottingham University
Nottingham, GB NG7-2RD

Thomas L Bucy 8905
Bucy Die Casting
633 South Glenwood Pl.
Burbank, CA 91506
818-843-5044

Bob Bybee 8904
Poly Peripherals
5011 Brougham Court
Stone Mountain, GA 30087
404-498-3556

William H Chaffee 8804
1301 S.E. Riverside Drive
Evansville, IN 47713
812-423-2145

Fred Chambers 8903
900 Glen OOak Drive
Fairfield, AL 35064
205-787-2200

James Clay 8804
243 Old Adobe Road
Los Gatos, CA 95030
408-371-3240

Teresa Conant 8806
Blair House Appartment # 502
8201 16th Street
Silver Spring, MD 20910
301-588-7042

F. Marc de Piolenc 8902
4618-1/2 Pendleton Street
San Diego, CA 92109
619-272-1725

August L Flassig, Jr. 8902
W9HXO
75 Hillcrest, O.D. 210
Portage, IN 46368
219-762-1632

E K (Chic) Fox 8906
116 Central Park South, 5B
New York, NY 10019
212-582-3524

J. Earl Gilbreath 8906
214 Mc Laws Street
Savannah, GA 31405
912-355-4415

Fred Gohlke 9900
Amateur Computer Group of NJ
Post Office Box 135
Scotch Plains, NJ 07076
201-969-3544

James Goodman 8905
Datacare of Memphis
Post Office Box 22908
Memphis, TN 38122-0908

Charles W Gross 8806
3731 Steve Drive
Marietta, GA 30064-1662
404-426-7356

Douglas Gurney 8804
316 Chrystan Court
Montgomery, AL 36109-3919
205-277-1269

Don Haywood 8904
4225 Table Mountain Place
Fort Collins, CO 80526
303-229-0494

Douglas A Henry, II 8806
30W 211 Claymore Lane
Naperville, IL 60540
312-355-6594

Susan Hockert 8806
Hockert Computime, Inc.
5300 Royal
Dallas, TX 75229
214-321-5319

Robert K Johnson 8902
5303 Luwana Drive, S. W.
Roanoke, VA 24018
703-774-3979

Robert L Jones 8906
ADirections, Inc.
3222 North High Street
Columbus, OH 43202
614-885-7728

Bob Kelso
108 Belle Chasse Drive 8903
Lafayette, LA 70506
318-981-1971

Ralph E. Kenyon, Jr. 9999
Abstract Systems, etc.
191 White Oaks Road
Williamstown, MA 01267-2256
413-458-8421

Robert B Kirkley 8805
707 Meadowick Drive
Baytown, TX 77521
713-427-5321

Al Levy 9900
The Stack, L.I.C.A.
P.O. Box 71
Hicksville, NY 11801
516-293-8368

Mitchell S. Lippman 8905
1295 Terrell Mill Rd. S101
Marietta, GA 30067
404-952-4152

George Little 8806
1850 Adanac Street Apt. 314
Vancouver British Columbia
Canada, V5L-2E3
604-255-2300

Kenneth Lowe 8903
5936 West Zina Circle
West Valley City, UT 84120
801-969-7736

Charles & Helen Mach 8804
709 Bowman
Irving, TX 75060
214-259-7964

George & Reva Mack 8906
8224 Ravinia Road
Fort Wayne, IN 46825-3431
219-489-6638

John Matelock 8906
155 Dale Avenue
Cambridge Sprgs., PA 16403
814-398-2795

Thomas E Miller 8805
1198 Azora Drive
Deltona, FL 32725
305-574-6361

Ronald C Moffatt 8905
126 Suellen Drive
Rochester, NY 14609-3221
716-482-1136

George D Montillon 8902
G D M Associates
351 Fleming Road
Cincinnati, OH 45215
513-761-1565

John R Neal 8903
Post Office Box 550127
Dallas, TX 75355
214-340-1464

Russ Nobbs 9002
Rings & Things
Post Office Box 450
Spokane, WA 99210-0450
509-624-8565

Sirous Parsaei 9999
PolyMorphic Systems
805-967-4423

Constantin Pavloff 8902
1726 Horn Avenue
Richland, WA 99352-2314
509-946-6553

James P. Purvis 8902
501 Northpointe Pkwy, #316
Jackson, MS 39211-2357
601-956-0668

Robert+Rose Quinn Prater 8805
Post Office Box 163356
Austin, TX 78716-3356
512-444-5229

Percy Roy 8904
11228 94th Street
Edmonton, Alberta
Canada, T5G-1G9
403-468-5831

James Salinger 8902
James Salinger & Associates
P.O. Box 37245
Cincinnati, OH 45222
513-531-3106

Douglas R Schirripa 8902
Post Office Box 127
Ionia, NY 14475-9708
716-657-7437

Robert L Schwartz 8902
Schwartz & Schwartz
906 Main St. #405
Cincinnati, OH 45202
513-241-3447

Tim Scully, Ph.D. 8903
Mendocino Microcomputers Inc
32191 Albion Ridge Rd.
Albion, CA 95410
707-937-5001

John G Sparti, D.O. 8805
Dallas Family Practice Ct
318 Main Street
Dallas, GA 30132
404-445-6140

R Conway Spitler 8806
943 North Oak Avenue
Fillmore, CA 93015
805-524-1351

Charles Steinhauser 8906
East Ponce de Leon Avenue
Stone Mountain, GA 30083
404-297-7579

Gary A. Sterling 8903
Rural Route 1, Box 220
Hedrick, IA 52563
515-661-2986

Robert C Stricklin 8902
Post Office Box 1931
Dallas, TX 75221-1931
214-528-3806

Robert A Teall 8806
Bob's Industrial Electronics
1720 North Sherman
North Platte, NE 69101
308-532-8017

Karl E Thomas 8902
Thomas Electric Company
145 Bond Street
Elk Grove, IL 60007-1218
312-956-6300

Charles A Thompson 8806
2909 Rosedale Avenue
Dallas, TX 75205-1532
214-368-8223

Len Thomsen 8804
Post Office Box 637
Surrey, British Columbia
Canada, V3T-5L9
604-590-0047

James J. Trahan 8901
OSO Investments
Post Office Box 217
Oxnard, CA 93032
805-487-2774

J. Reid Turnquist 8902
Post Office Box 2448
Dillon, CO 80435
303-468-2490

Richard Wagner 8805
The Centre Shepherd's Bush
5416 Gaston Ave
Dallas, TX 75214
214-823-2021

John J Warkentin 8805
2248 San Remo Court
Pittsburg, CA 94565
415-439-9407

William E. Wright 8805
Post Office Box 570368
Houston, TX 77257-0638
713-975-1188

Logic Families

by Bob Bybee

Percy Roy and others expressed an interest in replacing some of the Poly's ICs, with CMOS chips. You asked for some explanation of what can be replaced, so here I go!

Most of the parts on any Poly board are what's called TTL, for "transistor-transistor logic." These parts generally have numbers like 7400, 74LS00, or 74S00. Any part that begins with a 74 and ends with the same 2- or 3-digit number (in this case, 00) has the same LOGICAL function. All of the 74x00 parts listed above perform the function of a quadruple, 2-input NAND gate. But they differ in speed, power, input loading, output drive, and input/output voltage levels (even though they all run off of +5 volts).

If you're going to substitute one logic family for another, it pays to understand a little about the various families involved. Here is some background on each family member.

7400: The patriarch of the TTL clan. Since Poly boards are old, you may see some 7400 series stuff on them, but 7400 parts have mostly been superseded by 74LS00.

74L00: Low-power TTL. It was also slower than 7400, and is obsolete.

74S00: "Schottky" TTL, named for the inventor of the fast diodes which speed up the internal operation of these parts. They are much faster, use more power, and have higher input/output currents than most other TTL families.

74LS00: Low-power schottky. Uses the fast diodes, but in a lower-power design. The LS parts offer an excellent compromise between speed and power, and are probably the most common parts found in computers dating from Poly to today's systems. LS can generally replace 7400 parts anywhere.

All of the above parts are still TTL, which are relatively high-power parts. If you're trying to reduce power, you do need to go to a CMOS (Complementary Metal Oxide Semiconductor) part. CMOS parts generally use much lower power than TTL, but they are also not completely interchangable. Here are some CMOS families you may encounter.

74C00: Obsolete, slow, and not interchangeable with TTL.

74HC00: A little slower than TTL, but uses different voltage ranges for input, and so is not TTL-compatible.

74HCT00: Similar to HC, but especially designed to use TTL-compatible voltages. Despite the fact that it's a little slower than 74LS00 parts, most boards can use 74HCT parts as replacements. THIS IS THE

BEST CMOS PART TO USE AS TTL REPLACEMENTS.

There are also some newer, faster, and more advanced logic families available now, both in TTL and CMOS technologies. You're not as likely to encounter these, but here they are anyway.

74F00: "Fast" TTL. Very high speed, fairly high power. Often can be used in place of 74S00 parts.

74ALS00: "Advanced Low-Power Schottky" TTL. Not quite as fast as 74F, but close, and lower power. Also has low input loading and high output drive, which makes it a suitable (but expensive) substitute for 7400, 74LS, and sometimes S.

74AC00: "Advanced CMOS". Fast, low-power, and not TTL-compatible.

74ACT00: Same as AC, but designed to be TTL-compatible. But a warning: ACT parts are so fast, and have wide voltage swings, that they can cause noise problems on boards not designed for them.

In general, a part with a "C" in its name is CMOS, but unless it also says "T" (as in HCT), it's not TTL-compatible, and cannot be substituted for TTL.

One question that came up was: do all parts have to changed at the same time, when going to CMOS? Not if you use HCT. If you chose HC parts, you would have to change all parts at once (since HC and TTL parts do not have compatible voltage levels), but even that would be no guarantee that the board would work. Certain other components (RAMs, EPROMs, etc.) would not be compatible with the HC-family parts.

What would you gain by changing all your parts to CMOS? Well, the board would draw less power, therefore run cooler, and therefore your system might last longer. It won't make anything run faster, or get better answers from your accounting package.

When using CMOS parts, be extra careful - any IC can be damaged by static electricity, but CMOS is even more susceptible than TTL. Always touch a metal table, the Poly chassis, or SOMETHING, before touching an IC or a board.

[Bob can be reached at Poly Peripherals, 5011 Brougham Court, Stone Mountain, GA 30087, (404) 498-3556... Ed.]

Cover your Keyboard

On my desk sits a TI-59 programmable calculator, nested neatly on its PC-100C print cradle. Both are covered by a dust cover designed to fit perfectly over the two as a unit. On the computer table next to my desk sits my Keytronic 5151 Dvorak keyboard with its perfectly fitting dust cover. It has always annoyed me that I can't get a dust cover which is 'just the right size' for my Poly keyboard.

Well, this last month I took matters into my own hands and made a cover for my Poly Keyboard II. It took a little experimenting, but it came out pretty neat. I can't take all the credit, since I gave a keyboard to my mother and asked her to make one for me. Well, her cover fit more loosely than I wanted. After all, the Poly is only 13 and still needs room to grow. Sorry Mom, I couldn't resist that. I looked at what my mom had done and decided that I could do it too.

Well, I made my cover from a piece of thin leather I had purchased for another project. But, Naugahide or some similar material would be just as suitable. First, I traced the end of the keyboard; then I added a 1/4 inch border around the tracing for the top, front, and back of the tracing. This 1/4 inch extra provides material for the seam when it is sewn to the top of the cover. Next, I stretched a piece of aluminum foil over the top of the keyboard to get a pattern. It turned out that the span across the keyboard is just 10 inches. I am using Keyboard II, which is 13-3/4 inches wide, so I made a paper pattern 10 inches by 14-1/4 inches (adding 1/4 inch on each end for the seam). Next, I found a piece of thin leather and cut out the three peices to match the pattern. I was careful to make sure the end pieces were cut facing the correct direction. (In sewing, the "face" of the material is the surface which is to be exposed or seen when the piece is finished.)

At this point I got out my trusty sewing machine and tried to sew the end pieces to the cover along the 10 inch edges. It turned out that trying to sew around the rounded corner of these end pieces was nearly impossible. I remembered that notches are cut in materials where corners are sewn. So, I took a careful look at the end pieces and made right angle notches where the corners are. I also made 1/4 inch slits in the appropriate places on the edges of the top piece. It turns out that those slots are just 2-3/4 inches from one end and 1-1/4 inch from the other end. (See the pattern.)

Sew, (ha ha) here's how you can make your own cover for the Poly Keyboard. You will need a piece of material which is 10 inches wide and, depending upon whether you have Keyboard II, or Keyboard III, 18-1/4 or 21-3/4 inches long. Thin leather, or Naugahide, or any other suitable (to your

tastes) material will do.

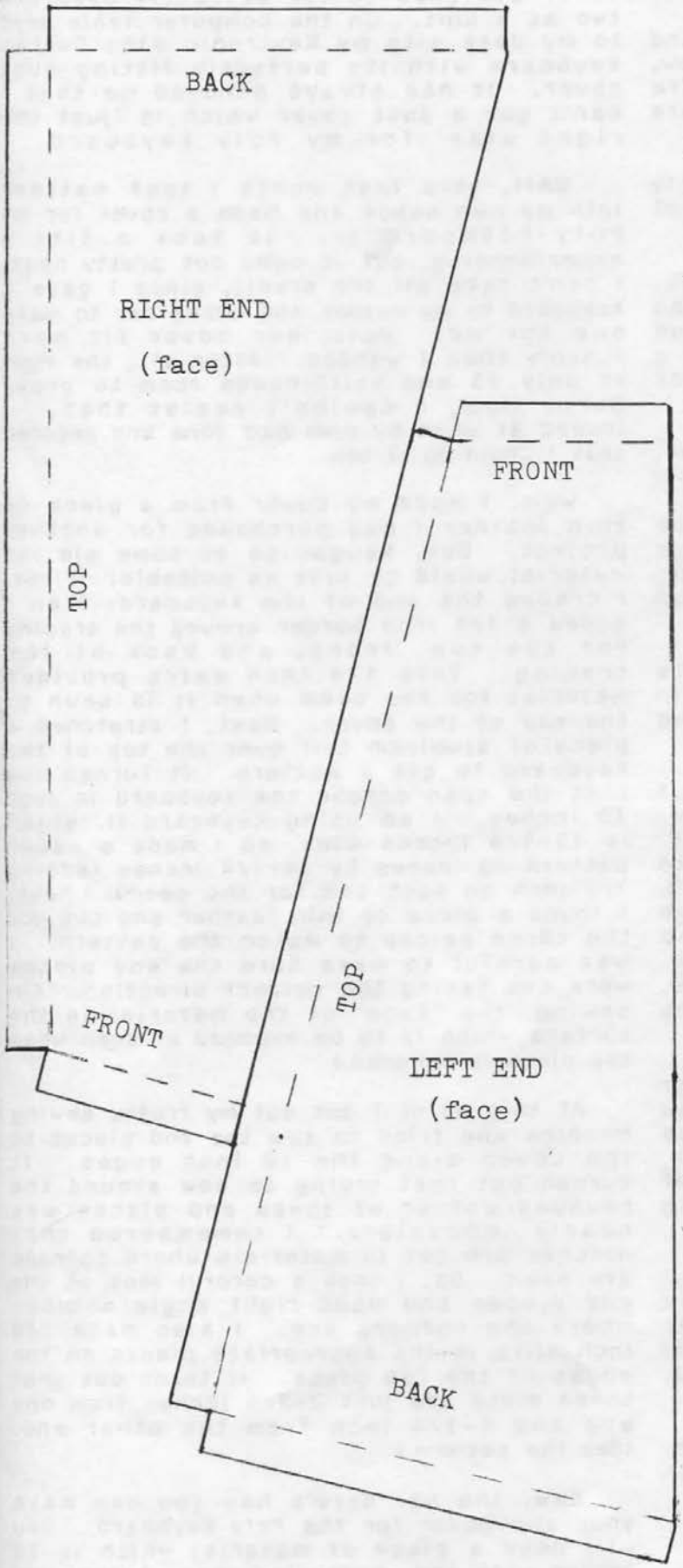


Figure 1.

First cut the top piece.

Keyboard II	10 by 14-1/4
Keyboard III	10 by 17-3/4

The remaining piece should be 10 inches by 4 inches. This will be enough to provide the two end pieces, but it is best to make a paper pattern for those pieces. Trace or cut out the pattern in figure 1. When we lay the two pattern pieces on the remaining fabric, be careful to orient them so that the resulting fabric pieces will have the proper side facing out. Figure 1 shows the proper orientation.

Next, lay out the top piece so that the 10 inch ends are to your right and left. You will be cutting 1/4 inch slots in the end 1-1/4 inches from the front and 2-3/4 inches from the back on both sides. See figure 2 for the location of these slots.

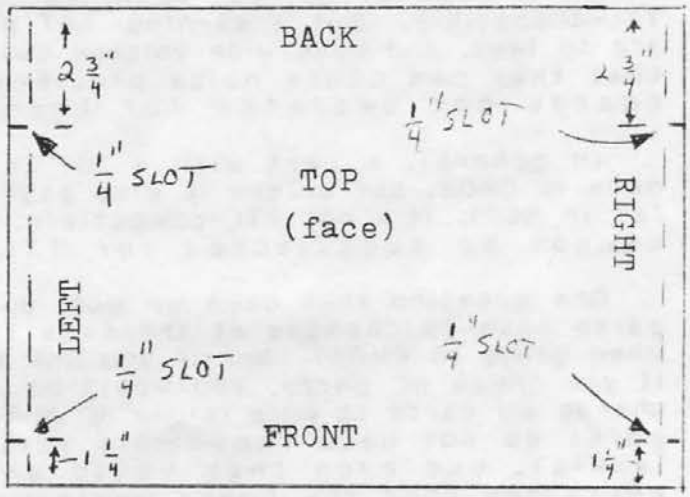


Figure 2.

Once these pieces are cut out, you can sew them together to make your cover. Place the top face down (with the 1-1/4 inch slots away from you) and place the left end piece on top of the top, with its face up. Align the 1-1/4 inch edge with the 1-1/4 inch portion of the top. See figure 3.

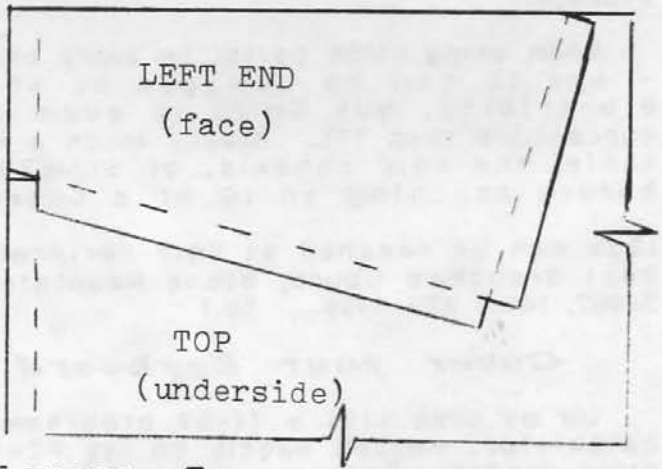


Figure 3.

Sew the two pieces together along the 1/4 inch margin line. Next, bend the end around so you can align the 2-3/4 inch edge with the 2-3/4 portion of the top; it will be necessary to fold the top over itself to do this. Sew these two portions together along the 1/4 inch margin. See figure 4.

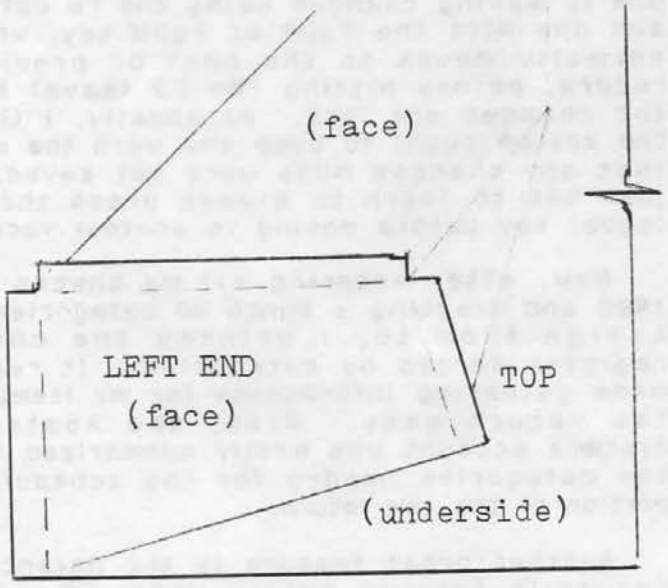


Figure 4.

Next, align the remaining 6 inch edge of the end with the remaining 6 inch edge of the top and sew them together along the 1/4 inch border. See figure 5.

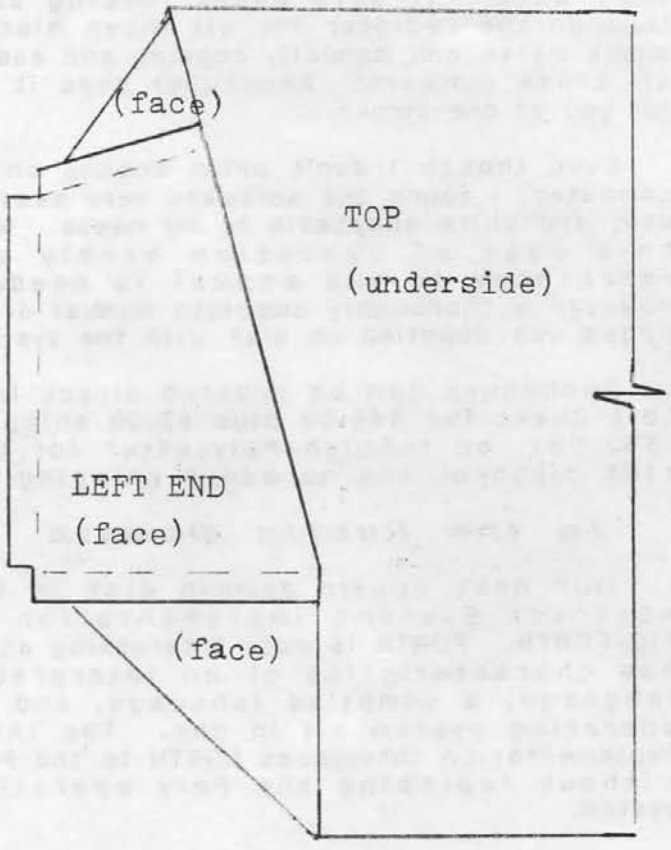


Figure 5.

The other end is just the mirror image of the process. I also used a zig-zag stich on the three exposed tabs on each end. To really finish the task with a classy ending, 1/4 inch seam binding should be affixed to all the protruding edges. Do each end first, bending the binding around the slots in the corners. The , starting in the middle of the back, attach one run of binding all around the bottom edge. With the finished product, once again, the Poly out-classes the clones.

Exchequer

I recently received a copy of a program for the IBM-PC and compatibles which is a check writing and checkbook accounting program. I tried out the program and found it extremely easy to use. While I don't use my computer to write checks, I do keep track of my accounts on it. Exchequer 2.0 is written by Left Coast Software, P. O. Box 160601, Cupertino, CA, 95016-0600, and retails for $49.95 plus $3.00 shipping.

According to Left Coast, Exchequer runs on any IBM PC, XT, AT, or PS/2 compatible computer running DOS 2.0 or higher and works with any graphics adapter. It requires about 230K of available memory and can run in a single floppy environment.

The "top level" organization is composed of the three main files that comprise an account. These consist of the check register itself, a file of memorized checks, and a file of categories which can be assigned to each check or transaction. (A transaction can be split by assigning parts to different categories.) The beauty of the system is that the working display of these files LOOKS LIKE a personal check register being displayed on the screen. It is familiar and easy to understand.

Exchequer makes maximum use of the function keys.

Each display shows the function key commands in one line at the bottom of the screen. (A built-in self-help facility.) Here's the function key "menu" structure which shows most of what the program can do. Each indentation is a sub-menu.

F1 - Quit (Save data files and return to DOS)
F2 - Setup
- Print Test Check
- Set Drive/Directory
- Set Today's Date
- Set Starting Check Number
- Modify Program Defaults
  - Default account name
  - Default drive/directory
  - changes between autosaves
  - Printer for reports
  - Printer for checks
- Modify Check Layout

F3 - Account
- Balance/Reconcile Account
- Open New account
- Load Existing Account
- Backup Account Files
- Modify Account Data
  - Account Name
  - Account Number
  - Account Description
  - Category Filename
  - Checks filename
  - Check Layout Data
  - (plus printing options)
- Roll Account Forward

F4 - Category Screen
F5 - Memorized checks Screen
F6 - Check Register
F7 - Delete Highlighted item
F8 - Modify Highlighted item
F9 - Add/Save
F10 - Reports
- Checkregister
- -Sorted by category
- -Sorted by payee
- -Single Category
- -Single Payee
- Uncleared Checks
- Category Totals
- Income/Expense Report
- List of Memorized Checks
- List of Categories

All reports can be selected to be sent to the Printer, the Screen, or to a File.

The overall design of the system is to allow the user to print checks and have each check added to the check register automatically when it is printed. It also allows tagging memorized checks for batch printing.

To print checks one selects the check screen with function key F5. Checks to be printed can be tagged with the space bar. Then one uses function key F10 to print the checks. The program will print each check in turn. When a check is printed it is automatically added to the check register. When no more tagged memorized checks are left, the program presents a blank check form to allow printing checks which have not been memorized. Memorized checks can include deposit and withdrawl transactions as well.

Now, I noted with pleasure that the categories screen can be brought up at almost any time and can even be updated by adding, deleting, or changing whenever it is brought up. So, I can create a new category even while I am writing checks. (The ESCape key returns to the previous process.) It is also possible to memorize a check while writing one.

In my case I was entering checks I had previously written and used the register screen as my base of operations. The addition process is simple because it automatically supplies the next check number and the last used date. Even changing the date is painless; entering a slash skips to the next date field.

I did encounter one minor annoyance. If one is making changes using the F8 option, and one hits the PgUP or PgDN key, which normally moves to the next or previous record, before hitting the F9 (save) key, the changes are lost. Personally, I think the system ought to beep and warn the user that any changes made were not saved. I just had to learn to always press the F9 (save) key before moving to another record.

Now, after entering all my checks for 1988 and creating a bunch of categories to assign them to, I printed the check register sorted by categories. It really made gathering information for my itemized tax return easy. Also, the Abstract Systems account was easily summarized into the categories needed for the schedule C portion of the tax return.

Another great feature is the Balance / Reconcile Account option under F3. One scanns through the register and uses the space bar to mark as cleared the checks and other transactions which have cleared the bank. You enter the bank statement balance and Exchequer will add up all the uncleared checks and deposits and prepare the figures needed to reconcile or balance the checkbook. It sure beats looking back through the register for all those missing check marks and manually copying and adding all those numbers! Exchequer does it all for you at one stroke.

Even though I don't print checks on my computer, I found the software very easy to use, and quite adaptable to my needs. With this ease of operation hardly any references to the manual is needed. However a thoroughly complete manual of 61 pages was supplied on disk with the system.

Exchequer can be ordered direct from Left Coast for $49.95 plus $3.00 shipping ($52.95), or through PolyLetter for $45 (10% discount and no added shipping.)

In the Public Domain

Our next public domain disk is the Abstract Systems implementation of FIG-FORTH. FORTH is most interesting as it has characteristics of an interpreted language, a compiled language, and an operating system all in one. The {AIS} implementation interfaces FORTH to the Poly without replacing the Poly operating system.

BASIC has a fixed number of keywords like "PRINT", "REM", "RUN", "LET", "FOR",

etc. Some of them may only be executed at the BASIC prompt (">") like "RUN"; some may only be executed within a running program like "NEXT", or "STOP"; and some may be executed within a program or from the prompt. For example PRINT may be executed in either way.

FORTH has a dictionary with a variable number of keywords. At any time one can create a new keyword with the colon definition. Once a new keyword or FORTH definition has been created, it can be used in any new definitions. That would be like each BASIC program name being added to the list of BASIC keywords.

```
Disk PGL-V-29 has 44 files on it, 18 free entries.
348 sectors in use, 0 sectors deleted, 2 sectors free.
 Size Name.
    4 SYSTEM.SY     3 Fmsg.OV    23 FORTH.GO     1 TEST.FX
    1 GO.TX         8 Fmsg.AS     5 FM.FX        4 8-QUEENS.FX
    5 VLIST.TX      2 KEY.FX      1 DEMO.TX      1 FIG-FORTH.TX
    7 DOC.TX        5 prin-for-msg.AS       2 prin-for-msg.GO
```

PGL-V-29 is an interpreter for the FIG-FORTH implementation which has been modified to use the PolyMophic Systems disk operating system. The LOAD command has been modified to allow LOADing a FORTH text source program from disk. The syntax is: LOAD File-name.FX (FX for FORTH TEXT). The system has been modified to provide an error message overlay, and the familiar Poly CTRL-W, CTRL-X routines are supported. Documentation on FORTH itself is not included, however is available from the FORTH Interest Group (address included). The details of the PolyMorphic Systems implementation are included. A demo application (computing a general quadratic equation) is included.

BugNotes

Abstract Systems BugNote 017.0 **April 11, 1983**

Sio.PS (for Printers 37 thru 42)

Sio.PS has a design bug in the input routine. Whenever the receive buffer is full, incoming characters are simply "thrown away". The Sio driver was written without any provision to drop DTR or RTS when the incoming character buffer is full. Sio does NOT program the 8251 USART to stop incoming data by dropping DTR or RTS. For hardware handshaking, the RTS line from the Poly must be connected to the Printers CTS line so the printer can be sent a 'busy' signal. Sio does not send a busy signal; it gets the character and throws it away.

What does this mean in practical terms? Programs which read data from printer devices such as two way terminals and printers with keyboards can lose incoming data. This is particularly true for high baud rates. Terminals which can transmit an entire screen of data at high baud rates will over-run the program reading the data from the printer driver (Sio). Characters will be lost. At 9600 baud, a machine language program (DownLoad.GO) which loops, simply reading a character and putting it in memory loses about 2/3rds of the data. Less is lost at lower baud-rates, with no loss at 1200 baud. Programs which are slower than a machine language downloader (BASIC or other high level language programs) will lose proportionally more data.

June 23, 1983

Abstract Systems has a modified version (sio.PS) which corrects this bug.

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 Printer
      HELP file for system command "Printer"

The "Printer" command selects a printer or executes a printer
sub-command.

Syntax is : "Printer <name or command>" (RETURN)

"Printer" lists commands and known printers.
      Commands: LOG  NOLOG  SET  SHOW
      Known printers: Null  Screen  File  prism  X2

"Printer <name>" connects the device called <name>.
      ("Null" and "Screen" are system devices.)
"Printer Null" disconnects all devices. Characters which would
      have gone to the printer are discarded.

"Printer Screen" connects the video screen. Characters sent
      to the printer will be displayed on the system screen.

"Printer LOG" records all screen transactions on the printer.
"Printer NOLOG" turns off Printer LOG.
"Printer SHOW" displays the current page parameter settings.
      Lines per page (form size):
      Characters per line (page width):
      Lines for TOP margin:
      Lines for BOTTOM margin:
      Offset for left EDGE:

"Printer SET" allows setting the page parameters.

The minimum size needed for the system to recognize this
command is "Pr". Sub-commands or names must be spelled out.
Example: To connect the system video screen, use "Pr Screen".
```

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. Noncommercial ads by subscribers are free.

FOR SALE: 400 4116 memory chips. $200 for the lot. Charles Steinhauser, ~~2213 Voorhees, Apt. 102, Redondo Beach, CA 90278.~~ See new Address - Page 4.

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

In This Issue

Coming Soon

Modems and Communications software; More: BASIC for Beginners, System Programmers Notes, Help, BugNotes, Public Domain Software, etc.

Questions

Can you find and answer the questions asked in this issue? Send your answers and requests in.

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

Address Correction Requested

FIRST CLASS MAIL

Ralph E. Kenyon, Jr. EXP: 9999
Abstract Systems, etc. 184
191 White Oaks Road
Williamstown, MA 01267-2256

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 ads are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* are available at the same price as the previous subscription rate. (US $15.00 yr., Canada $18.00 yr., Overseas $20.00 yr., payabh% in US dollars to Ralph Kenyon.) Individual issues are also available ($3.50, $4.00, $5.00).

PolyLetter

Editorial

In closing their doors, PolyMorphic Systems has been liquidating their assetts. I bought a large box of used disks, and can offer them at $.50 each. On some of the disks, there were some goodies, including the Poly Meta compiler and a C like compiler called the Orange compiler. Sirous has consented to put these in the public domain, so they will be organized and placed in future PGL volumes. If anyone is in a hurry for anything call me and I'll speed up the process. I still have lots of stuff to go through, but it's mostly old system disk. Speaking of the PGL, I have condensed the listings of all the PGL disks and included that here for review. It has been some time since we reviewed what's available.

The Copyright issue has been in the news lately, and I have had some more thoughts on that also.

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. Noncommercial ads by subscribers are free.

FOR SALE: 400 4116 memory chips. $200 for the lot. Charles Steinhauser, 404-297-7579

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.

My Favorite DOS Programs

by Al Levy

Thanks for PolyLetter. You are certainly doing a great job! In response to your request for favorite programs under DOS:

Program # 1 - ProComm - Modem
Program # 2 - Quicken - Checkbook
Program # 3 - Dataflex - Database
Program # 4 - Where - Utility
Program # 5 - Browse - Utility
Program # 6 - Ventura - DeskTop Publishing
Program # 7 - Norton Utilities - Utilities
Program # 8 - RENDIR - Utility
Program # 9 - STRIP - Utility

Program #1 - PROCOMM - Shareware - available to try for $2. I like Procomm for file transfers. It works great with the poly at 1200 baud, and sometimes at faster speeds. On the I.B.M. side you use PgDn for receiving a file and on the Poly side use the PRint command. Step one is to set up a printer on the Poly using Setup

```
$Setup
New PC
Similar to a Diablo? N
Understand Form Feeds? Y
Understand TAB characters? Y
Speed of printer (in BAUD)? 1200
Blocking type device? N
ASCII code for PAD character? 0
Number of Pads after CR? 0
```

```
Number of Pads after LF? 0
Number of Pads after TAB? 0
Number of Pads after BS? 0

Default page parameters
Lines per page (form size) 255
Characters per line (page width) 255
Lines for TOP margin? 0
Lines for BOTTOM margin? 0
Offset for LEFT edge? 0
E (to exit setup)

Pr PC - Exec command to setup this printer
PR <DRIVE>FILENAME
```

I am assuming that you have a null modem cable connecting the Poly serial port and the I.B.M. serial port.

Program #2 - Quicken - a cute, easy to use check book program (commercial product).

Program #3 - Dataflex - Heavy duty database - cost $700 - I've been using this for six years. Aint nothin like it on the market. Works similar to poly programs. Wrote stuff like this in BASIC on the Poly. It took months. Dataflex - 1/2 hour maybe. Really relational and because of the FLEXability I might spend months designing a system. Reports are automaticly generated and you can use up to TEN indices. Think of our pityful MaiList with just one. When I wrote for the poly I created indexes for various fields in a database and used Ralph Kenyon's Heapsort when data was entered or stored. In dataflex this is all automatic (no coding).

Program #4 - Where - A Public Domain Program which finds files on a hard drive (when you forgot where you stuffed them) or, displays duplicates etc. Very much like Poly's "WHERE.GO".

Program #5 - BROWSE - Also PD - Allows one to view a file on the screen and do Page Up, Page Down, use the cursor keys etc. Unlike the Poly, IBM DOS scrolls to the end of the file unless you use "More" (UGH!) or hit CTRL/S.

Program #6 - Ventura DeskTop Publishing. Here is where advanced technology shines. I edit a 40 page NewsLetter for LICA (Long Island Computer Association) once each month. Text Files are read into "Pages". I usually type the text on the Poly and "Port" it over to the I.B.M. clone via a null modem and ProComm. I have an escape library on the poly to put in the proper commands to format each paragraph. This saves a lot of wrist action (you would have to Point & Shoot with a mouse otherwise).

Program # 7 - Norton Utilities - too many features to go into here but well worth the money. A Super "Szap.GO" and "Recover" + "Undelete" and plenty of other good things.

Program # 8 - RENDIR - Public Domain - The Geniusi at MicroSoft who wrote DOS allow you to Rename a file, but not a directory. This PD program gives you that ability.

Program # 9 - STRIP - strips out the high bit and creates an ASCII file from a WordStar file. (PD)

Since I am the librarian for LICA's ShareWare and Public Domain disks, I can let any person who subscribes to PolyLetter have a copy of the catalog disks at $2.00 each (+ shipping). Write if you are interested.

Now for the commercial:

I can transfer all data from Poly To I.B.M., MacIntosh, Apple, or CP/M format. If you are using Mailist or some other Database I can write the software to run on I.B.M. Call (516) 293-8368 if you are interested.

What's Copyrightable Anyway?

by Ralph Kenyon

Look and Feel law suits have begun to appear regularly in the computer software industry. Lotus is suing companies which produce products having a similar appearance to their 1-2-3. According to the Boston Business Journal on March 13, 1989, Richard D. Bezjian, president of Mosiac Software, was quoted as saying "Lotus' whole argument is based on a single line. They're [Lotus] saying you can't used the word 'move', the word 'worksheet', [or] the word 'file'."

The Journal also reported that a federal judge ruled in favor of Massachusetts Technologies, Inc, (MTI) who was suing two competitors for "copying its screen display". According to the Journal, pervious cases hinged upon the copying of source code, while this case depended upon the "look and feel" of the software package based upon its screen displays.

The question of what is and what is not piracy has become quite controvertial, with large companies claiming the "look and feel" of their software should be the determining factor, while small upstarts claim that such a position inhibits competition.

It's beginning to look to me like the copyright laws are not the adequate vehicle for resolving these issues. The original model of the Copyright approach held that the program source code was copyrightable and protected from unauthorized copying. True to that model, scholars would be entitled to make one copy for research purposes. Since the relation between source code and object code is essentially

fixed, copyright protection was soon extended to object code. Byte for byte copying was clearly piracy.

Well, I agree that there are some significant issues involved. They center on the notion of intellectual property and the equity principle that a person ought to receive a fair reward for his or her own work.

The patent laws give an inventor the exclusive right to develop and profit from an invention for a certain period, unless the inventor gives up that right in certain ways. But, he/she is treated as being the owner of the right and may sell or transfer it. This "intellectual property" concept is being applied in the area of Software, but by using the Copyright laws instead of the patent laws. How come? Well, I conjecture that it is cheap and easy to register a copyright ($10), and a "common law copyright" need not even be registered. But, getting a Patent is another thing. In getting a patent, an extensive search of existing patents is conducted and a patent denied if one already exists for the proposed invention. There are other reasons for denial, as one cannot patent an idea which is the common property of the culture. It must be a new idea, and it must be first to be patentable. Once a patent is granted, however, the product (device) is registered and just what aspect of it is specifically new is protected from copy infringement by legal penalties, which often include giving to the inventor all earnings a pirate has made from the device.

By patent laws, an algorithm implemented in a device would be patentable, and an improvement in the algorithm which was implemented in a different device would also be patentable.

So, I conjecture that sofware authors took the quick and cheap way out by choosing to avail themselves of the copyright laws instead of the patent laws. Think about it for a minute -- Copyright laws protect an individual from someone copying your printed or recorded work and selling it again without your permission. As I understand it (and I may turn out to be wrong) copyright laws allow copies to be made WITHOUT PERMISSION in the case of a single copy by a scholar for academic purposes. They may also make extract copies of sufficiently short pieces of the quoted work and include the quotes in new published works.

In the case of Music, copyright has been extended to reproducing the work in any form, including singing it. A musician is supposed to get a royalty payment anytime another musician sings or plays the first musician's song or melody. These "volatile" copies perish immediately, and enforcing these copyright laws is a nightmare. ASCAP (the Association of Composers and Performers) has many representatives going around to clubs where live or recorded entertainment is featured and putting the bite on the proprietor to collect royalty fees for any songs played. On the other hand, Persons in their own homes, or in their private, non-commercial gatherings are left alone. Songs may be reproduced without limit for private, non-commercial, consumption, but may not be reproduced for commercial consumption without the payment of royalty fees.

Now comes software. In truth, every time you run a program, you have just made a (volatile) copy of it in memory (not unlike singing a song) for immediate use and consumption. Making a backup copy of a program, like xeroxing your sheet music, seems to be ok, unless you do one of two things with it. If you sell the copy you violate copyright laws, and principles of equity by profiting from anothers work. If you give the copy away you also violate copyright laws and principles of equity by depriving the other of potential income from the sale they otherwise might have made.

These laws seem fairly adequate for exact copies of software source code. But what about cases where the code differs? Well, taking someone elses work and making minor changes to it reqires that the original author get the majority of benifit from any subsequent sales. Extensive redesign of a software package using the original software function as a model to produce a compatible product, but with independent effort seems, from the point of equity, to be something that the original package designer should not have the lion's share of benifit from. This is a grey area which is well covered in Patent law structure, but not considered in the copyright law structure. The patent office has an examining board which makes the decision and either a new patent is issued, in which case the new inventor gets the entire reward, or a new patent is not issued, in which case the new inventor does not get any reward entitlement (the original inventor is protected).

So, in the patent environment big money is spent up front to make a determination, while in the copyright environment the issues are resolved by big money after the fact in the form of lawsuits. In both cases, the advantage is held by those with the big money.

Well, the little guys have set the stage by appealing to the copyright laws in these intellectual property issues, and have begun to feel the ambiguity in the process

as issues surface which don't properly seem to fit into the existing legal structure.

The copyright laws allow one to design a printed layout and then prevent anyone else from selling it. But, the copyright law system does not provide a way of checking to make sure that printed layout is an original product. The benifit of the doubt is assumed and the burden of proof falls to any challengers of a copyrighed form. Ok, the designers of software have begun to take the position that one of these forms being presented on a video display terminal is copyrightable. So, copyright protection is being sought for a portion of a software package, the display interface layout. But, there is a motive behind this effort.

A new software product is released. Some hot-shot programmer tries it out and gets fed up with user-unfriendliness, waiting, and a host of other annoyances in terms of inputting data into the display form. This programmer says to him/her self, "Ha! I can do better than this." and sets out to write a system which is functionally equivelent, with the same data screen layouts, but with a much friendler user interface, and a much more efficient internal engine. So, he goes to market with it claiming his/hers is better than the original. Mind you, this programmer has never seen the source code for the original system, and has written the second system completely from scratch, but has incorporated the organization scheme for data entry into the program.

Soon the original product manufacturer sees sales dropping and gets a copy of the new product, and sees that it runs rings around their original product. Do they go back to the drawing board and improve their own? Nope; they slap a lawsuit on the clone product producer. If we are lucky the clone product producer wins, but not befor legal fees have dried up the funds for producing the product and we end up with a legitimate but unavailabel product. A second outcome good for the consumer, is the clone producer sells his product to the original producer and the clumsy old product disappears. But what usually happens is that the hot-shot programmer, being of limited funds, settles for much less than is equitable; then the clone disappears, and the original product continues to frustrate users. A byproduct of this is increasing the credibility of look and feel lawsuits.

The "look and feel" lawsuits take the position of the touring machine test. If it looks like a lamb, smells like a lamb and feels like one, then it is one. It doesn't matter whether it's suffed, live, or made of plaster. A lamb is a lamb is a lamb. It doesn't matter that philosophers have poked the touring machine test full of holes, and the nature of something cannot be determined by appearances. The big boys have learned that lawsuits are an effective way of affecting the balance sheet. Well, how much significance is there to similarity of appearance? Is is purely an arbitrary set of choices? I think not. Consider the Space Shuttle.

Everyone looks at the Soviet Shuttle and says "Boy, doesn't that look familiar; did they just copy ours?" Well, NASA and the physicists shrug off the similarity by admitting that the Soviets probably did use some copied technology, but the main point of the matter is the fact that the laws of physics are the same for the Soviets as they are for us. And it's the laws of physics which determine what the shape of the shuttle must be. The overall "look and feel" of a shuttle is the same regardless of who designs it; a shuttle is a shuttle is a shuttle, and its apearance is determined by its function.

Well, communications are also subject to laws, and the laws affecting communication determine what shape certain classes of messages must have. A person spends tens of thousands of dollars in getting an education. Part of that education is learning what the "experts" agree is the "correct" way to record certain kinds of information. Certified Public Accountants, in particular, have exacting specifications for exactly how certain reports must "look and feel" to be correctly identified as "acceptable". Managers of companies learn these rules so well that they think in terms of the balance sheet format. We understand when we are making a profit in terms of whether the "bottom line" is in the black or not.

Any piece of software which proports to be a tool for managers to use in getting a hold on the company position had better conform to the standard ways of presenting the data. Our paradigms are common to all of us and are public property. To present information in the shape of one of those paradigms is a necessary requirement for effective communication. We cannot "screw with" the word order in our sentences and expect to be understood. True, there are limited execptions. We tolerate babies in the early period of their learning, but not for long; we soon correct them by insisting that they use the standard formats.

In business communications no less is true; the presentation screens of business analysis software must conform to the accepted models for communication to be effective. But, those standard formats for data presentation, be they layed out the printed page, as in quarterly reports, or presented on a computer screen, as in LOTUS

1-2-3, must be conformed to.

The order of and placement of the individual figures in the presentation display format, and the words that describe what is presented, as well as the words that describe the options the presentee may take are all dictated by the relatively rigid standards which come from CPA's, the IRS, and other authorities.

One doesn't succeed in business unless one learns what the standard models are and how to understand the condition of the business using the information in these standard models. Look how pervasive the phrase "the bottom line" has become in our culture. There is no real option. The bottom line of the display must contain the information which we now call "the bottom line". There is no other effective communication choice.

Fords and Chevies don't have their brake and gas pedals on opposite sides. There is a standard which all must adhere to. While it is true that the standard may be somewhat arbitrary, it is, never the less, there, and must be responded to. What, on the other hand, is connected to the brake and gas pedal is an entirely different story. More efficient engines, faster response times, turning ratio's, etc, are all parameters by which automobile performance is measured and compared; pick up any issuer of Car and Driver.

Software which conforms to the only (unwritten?) business information presentation standard in existence and which uses the the correct terms in the language to describe the display and the user's options can also be compared by various performance parameters. How much computer memory is used by the system, how fast it responds, how user friendly mistakes are processed, etc., are all parameters by which software performance is measured and compared.

All this "look and feel" litigation is just plain rediculous. Fords, Chevy's, and Chrystler's all have a similar "look and feel" because the manufacturers are constrained by the same physical laws, and the limited variation among human beings, as well as the (albit arbitrary) same standards for how people use automobiles. Form is determined by function.

No less is true in software design. The form of the communications interface between people is constrained by standards, physical constraints, and how the user needs the information presented so that that user may understand and respond properly to the information.

How can these companies copyright a format that has been in use on paper by CPA's for decades? Communications interface standards in human communication is part of our public heritage. We communicate effectively to the extent that we learn and adopt the standards, and no one has a monopoly on any of these standards.

Human Factors Engineering has begun to extend its domain to how humans interface to information processing systems. Professor Jefferson Koonce of the University of Massachusetts at Amherst stated that these presentation systems are constrained by the experts' [users'] models of the process. "In designing the system we first find out what the model is the expert has in his head for understanding the process. Then we design the instrumentation to display the information in the manner that the experts' model calls for."

The movies depict elaborate control panels in nuclear power plants which show a miniature picture of the plant itself. Nuclear power plant operators cary a miniature picture of the plant in their minds; the display corresponds directly to that mental picture. My own experience in Submarines conforms to this requirement. Display panels show the position of the planes and the attitude of the submarine, which are the significant pieces of information when it comes to trim, which is the Diving Officer prime responsibility. The information display system must conform to this model to be effective. Similarly, Airplanes have attitude display instruments which fit the aviator's mental picture of the plane in the sky.

A spreadsheet program must present a picture of the company which strictly conforms to the manager's mental picture of the company, and that picture is determined by the report formats which have been used by CPA's and the IRS for decades. The idea that a spreadsheet program designer can copyright this standard communications interface format is overextending the concept of copyright.

It cannot be that the standard elements of communication in a particular field are subject to copyright. If we allow that, soon no one will be able to repeat another's sentence without giving copyright notice.

In the Public Domain

Our next public domain disk is full of Abstract Systems utility programs, including many source files.

Disk PGL-V-30 has 35 files on it, 26 free entries.
350 sectors in use, 0 sectors deleted, 0 sectors free.

Size	Name.	Size	Name.	Size	Name.	Size	Name.
7	Select.GO	50	Select.AS	7	VSelect.GO	1	HELP.GO
4	HF.DX	18	VSelect.DC	2	Search.GO	17	Search.AS
19	MACROS.SY	5	ASCII.SY	8	SYSTEM.SY	3	Replace.GO
11	Replace.AS	4	Replace.DC	2	Restore.GO	11	Restore.AS
3	Rebuild.GO	9	Rebuild.AS	4	HELP.AS	2	ReStart.AS
1	ReStart.GO	5	Move.GO	29	Move.AS	1	Bell.GO
1	Bell.AS	1	PANEL.GO	1	PANEL.AS	5	VOL.AS
1	VOL.GO	3	pram.GO	15	pram.AS	7	Loop.AS
1	Loop.GO	1	Abort.GO	3	Abort.AS		

Abort.GO is a program to use in a command file which checks for a key-stroke before going on to the next command. This program allows setting up an "infinite loop" command file which calls itself as the last command. Abort allows an orderly way to exit the command file by checking the keyboard buffer to see if any key has been struck. If no key has been struck, Abort.GO continues and allows command file mode to continue. If a key has been struck it ends command file mode. Abort.GO does not alter USER or system memory, so can be used in conjunction with interrupted BASIC programs.

Bell.GO is a program for use as the last command in a command file. It sends a ASCII 7 (the bell character) to the printer every 1/4 second to let the operator know that a long process has completed.

HELP.GO is a program that searches for a text file of help information and displays it on the screen. The help files may be on any drive.

Loop.GO is for use in a recursive command file. The first time it is executed, it saves the count in system memory. Each time it is executed again, it Counts down the number of times the command file has executed and exits when the count gets to zero. Loop.GO loads and executes in the directory area so as not to change any user or system memory. Loop should be the first command in the file, and the last command in the file should call the file again.

Move.GO is a utility program for moving a file from one location to another. It combines COPY and DELETE in one operation. In DISABLED mode it won't move system files. In ENABLED move it will move system files. The wild card "*" may be used for destination file of the same name. A handy utility for rearranging disk space which is much faster than using FUTIL when only a few files are to be moved. Move can be invoked with file names on the command line, or in a prompted loop (similar to SCOPY).

PANEL.GO is a program which simulates the CTRL-Z interrupt and invokes the front panel. It can be used to bring up the front panel from a command file.

pram.GO sets the printer parameters from the keyboard. It loads in the directory so as not to change the user program. pram.GO can run fully menu driven like the Printer SET command or can be run with the parameters on the command line.

Rebuild.GO is a program to enter names in a directory. It was designed to help with rebuilding a directory.

Replace.GO is a system developer's utility which replaces one program with an updated version having the same name and size (from another drive). Replace.GO copies the new version into the same disk area used by the old version. This program is ideal for installing modified versions of system overlays or other programs which need to be near the directory (to minimize look-up time). It is also useful for replacing a crashed program with a good copy from the backup disk. (Saves a lot of copy and pack operations.)

ReStart is a program that resumes an aborted command file. It assumes that a command file has been started and was aborted (Cmdf Abort). After "ReStart" is keyed in, the next command in the command file will be executed and command file mode will continue. ReStart only works from Exec. If no command file was active, ReStart returns to Exec. ReStart loads and executes in the directory so as not to affect other system and user areas. Restart cannot restore programs which take control of the CTRL-Y interrupt. In only restores the command file mode flag and allows continuing the command file with the next command. Restart.GO does not alter USER or system memory, so can be used in conjunction with interrupted USER programs.

Restore.GO is a disk system utility which reads and rewrites an entire diskett. Stray magnetic fields may cause a gradual weakening of the magnetic pattern written on the diskette. Sectors which have not been re-written in a long time may be subject to a gradual loss of data. Restore.GO reads and rewrites all used data areas, restoring the magnetic pattern to full strength. It cannot recover areas which have been dammaged, or are otherwise unreadable by the system. The starting sector may be specified. Restore exits on ESC or on an error.

Search.GO is a program to search text files for lines containing with the given text. It asks for a file name, and a search string. It then searches the file for lines containing the supplied string, and displays the matching lines on the screen.

Select.GO is a general file utility program which searches an input data file of fixed length records for a specified character

string. It asks for all necessary inputs. The output is versitile... you can select a copy of the data record containing the match, or just it's position in the file. You also have the choice of output to the Screen, the Printer or to create an output data file containing the output. The output files are compatable with BASIC.

VOL.GO reads and displays the disk name for the selected drive. If no drive number is given, VOL process the drive in SYSRES. If VOL.GO gets an error from Dio, it displays the error in place of the name. Example for "No disk or door open." VOL.GO loads and executes in the directory to prevent trashing user programs, so can be used from an interrupted program.

VSelect.GO is a general file utility program which searches an input data file of fixed length records which also have variable length fields (marked by carriage returns) within records for a specified character string within the field. VSelect.GO is an enhanced version of Select.GO. The ability to handle variable fields, such as might be created with the BASIC WRITE command, is the only diference between Select.GO and VSelect.GO. Select.GO

PolyGlot Library

Disk of the month APR-80 has 5 files on it.
2 COUNT.GO 1 CONTROL-U.GO 7 CALENDAR.BS
4 READ-THIS.TX

Disk of the month AUG-80 has 33 files on it.
2 POP.GO 8 SZAP.GO 1 Cursor.GO
8 COPYALL.BS 4 MOVE.BS 2 ROOM.GO
3 POKE.BS 5 COPY-SUB-DIR.BS 2 Cursor.DT
1 BIG-LINE.BS 2 MIRROR.GO 3 DUP.GO

Disk of the month JAN-81 has 25 files on it.
37 SORT-DEMO.BS 44 DEMO-STRINGS.DT 6 MAZE.GO
17 READABILITY.BS 38 GENE.BS 44 HOME-INVENTORY.BS

Disk of the month MAR-81 has 33 files on it.
5 TABBER.BS 7 FNTIMER.BS 7 PEEK-DUMP.BS
3 Tran.OV 7 TRAN-DOC.TX 2 TRAN-TEST.BS

Disk of the month MAY-81 has 32 files on it.
13 BIORHYTHM.BS 35 BATTLESHIP.BS 15 HANGMAN.BS
2 SPIRAL.BS 1 ART.BS 22 BACKGAMMON.BS
19 HANGMAN-II.BS 31 ROULLETTE.BS 10 FROGS.BS

Disk of the month JUL-81 has 27 files on it.
3 READ.GO 5 COUNT.GO 5 FLIES.BS 12 INPUT.BS

Disk of the month DEC-81 has 31 files on it.
17 SLOT.BS 22 BACKGAMMON.BS 9 MOON-LANDER.BS
6 ARTIL.BS 22 Sex-Appeal.BS

Disk of the month MAR-82 has 15 files on it.
6 Letter.TX 12 Leaflet.TX 132 VM.VM
33 INITIAL.GO 2 FETCH.GO 1 PUNCH.GO
7 Bchr.OV 7 BCHR-DEMO.TX 1 bchr-demo.GO
2 BCHR-TEST.BS

Disk of the month JUL-82 has 31 files on it.
6 SDIR.GO 3 FIND.GO 16 FIND-&-SDIR-INFO.TX
5 DX.GO 1 FPL.GO 3 COMMENTS-BOWLING.TX
44 BOWLING.BS 47 TEXT-TRAN.BS 18 TEXT-TRAN-INFO.TX
3 TEST.TX 2 TEST2.TX 23 MASTERMIND.BS
1 CHANGE.GO 2 July.XD 4 MasterMind-Instructions.DT

Disk of the month MAR-83 has 12 files on it.
38 DATA-SORT.BS 36 ADD-A-BASE.BS 82 DATABASE.TX
1 CREATE-IT.TX 1 DIRECTORY.TP 1 DIRECTORY.DR
94 DATA-ENTRY.BS 5 BASES.GO 6 SCAN.GO
2 SNIFFALL.GO 4 ADDRESS.DX

Disk of the month JUL-83 has 12 files on it.
73 DIS80-SRC.TX 19 DIS80.GO 20 DIS80-INFO.TX
8 SYSTEM.SY 1 ERROR.GO 1 BERR.GO
6 MKDIR.GO 12 MKDIR-SRC.TX 10 ERROR-SRC.TX
10 BERR-SRC.TX

Disk of the month NOV-84 has 14 files on it.
9 CNT.GO 5 MERGE.BS 8 GEMSET-SRC.TX
2 GEMSET.GO 1 STRIP.CM 10 DDBLIST-INFO.TX
15 DDBLIST-SRC.TX 3 DDBLIST.GO 14 SETPR-SRC.TX
23 SETPR 1 LABE.TX

Disk PGL-V-01 has 27 files on it, 347 sectors in use.
50 STARTREK.BS 17 GALAXY.BS 15 GALAXY-INST.BS
2 USER'S-INST.TX 21 SURVIVOR.BS 15 CAI.BS
28 STARWARS.BS 22 BACKGAMMON.BS 16 HANGMAN.BS
28 DOCTOR-ELIZA.BS 23 LUNAR-LANDER.BS

Disk PGL-V-02 has 27 files on it, 251 sectors in use.
16 DOCUMENT.TX 49 INVENTORY.BS 13 STATUS.BS
4 CREATE.BS 11 STOCK.BS 9 INVENTORY1.DT
9 INVENTORY2.DT 2 DISCLAIMER.TX

Disk PGL-V-03 has 30 files on it, 349 sectors in use.
19 DECISION.BS 18 VENTURE.BS 4 DEPRECIATION.BS
31 ANNUITY.BS 5 SAVINGS.BS 26 INVEST.BS
22 LOANS.BS 6 INT78CRT.BS 7 STOCK.BS
7 PUTS&CALLS.BS 16 MLS.BS

Disk PGL-V-04 has 34 files on it, 350 sectors in use.
25 PERFIN.BS 8 CHECKBOOK-REC.BS 3 INTEREST-RULE-78.BS
5 T-BILL.BS 6 IRR.BS 3 FV-INVESTMENT.BS
10 FMRR.BS 15 FS-RA.BS 38 APARTMENT.BS
21 FUTIL.BS 7 CALENDAR.BS 11 FileSort.BS
3 FileSort76.BS 5 Calendar.BS 2 DIR-CREATE.BS

Disk PGL-V-05 has 29 files on it, 350 sectors in use.
7 M-L-R.BS 9 CPM.BS 11 PERT.BS
57 STATPACK.BS 18 REGRESSION.BS 23 L-REGRES(DEMO).BS
30 Plotter.BS 2 C(n:m).BS 3 N!.BS
2 TIMER.BS

Disk PGL-V-06 has 34 files on it, 350 sectors in use.
52 INVENTORY.BS 14 INVENTORY1.DT 40 DisAsmb31.BS
1 DM31H.BS 16 8080-INST.TX 10 DisAsmb.DC
8 Szap.GO 5 SCOPY.GO 2 SPACE.GO
2 CLEAN.GO 5 COMPARE.GO 4 DUMP.GO
1 DUMP.DC 1 Unsys.GO 1 scr.GO

Disk PGL-V-07 has 35 files on it, 349 sectors in use.
76 FORT.GO 13 FLOAD.GO 16 FORTLIB.TX
2 SAMPLE-1.FN 8 SAMPLE-2.FN 9 SAMPLE-3.FN
31 FORT-2^N.GO 9 PLOT.FN 1 FORT-DRIVE.DC
1 WARM.FN 10 SORT-EMPLOYEE.FN 2 EMPLOYEE.BS
2 INPUT-OUTPUT.DC 2 ALLOCATION.DC 4 DIRECTIVES.DC

```
   5 EXECUTION.DC      2 TEST.FN          1 g.TX
  21 TEST.HX          26 TEST.GO          (FORTRAN COMPILER)

Disk PGL-V-08 has 13 files on it, 350 sectors in use.
  70 MAILER.AS   14 RDE.AS            92 TERMINAL-FC.AS
  20 HAYES.AS    23 PROGRAM-LOAD.AS   17 TAPE-LOAD.AS
  52 MM100.AS    16 Space.AS          22 SPACE.TX
   5 Wait.AS      3 CMDF.AS            4 Tweak.AS

Disk PGL-V-09 has 20 files on it, 348 sectors in use.
   1 SCREEN-PROTECTED-DEMO.BS      4 MORSE-CODE.BS
   1 BASIC-USER-FUNCTION-TABS.BS   5 TELETYPE-CODE.BS
  34 LEMONADE-STAND.BS             2 ACCURATE-DIVIDER.BS
  57 STAR-LANES.BS                14 BIORHYTHM-2.BS
  58 SUN-RISE/SUN-SET.BS          53 MAJOR-MINOR-FINDER.BS
  17 SUN-RISE/SUN-SET.DC          21 SEAWAR.BS
   9 NUMBER-CONVERSION.BS          7 SYSTEM-SORT-PRINT.BS
   3 SORT-BUBBLE.BS                5 BASIC-USER-FUNCTIONS.BS
   6 SORT-HEAP.BS                 15 BIORHYTHM-1.BS
  14 HORSE-RACE.BS                18 INDUSTRIAL-SIMULATION.BS

Disk PGL-V-10 has 33 files on it, 350 sectors in use.
  22 RETIRED-PAY-FEDERAL.BS       13 LIFE-CYCLE-COSTS.BS
  17 Sorts.BS                     15 CAI-808.BS
   4 ADDITION.BS                   4 BASIC-USER-FUNCTIONS-1.BS
   8 SORT-CREATIVE-COMPUTING.BS    9 COPY-SUB-DIR-1.BS
   7 FILESORT-0.BS                 3 TOWERS-OF-HANOI.BS
   2 WONDEROUS-NUMBERS.BS         12 REVERSE.BS
   9 TOWERS-OF-HANOI-GRAPHICS.BS   3 PRINTER-COMMANDS.BS
  32 CRYPTOGRAMS.DT                9 CRYPTOGAME.BS
  20 BIG-LETTERS.BS                6 SHUFFLER.BS
  13 FORMAT.BS                     3 PROOF.BS
  25 ADDRESS-LISTS.BS              2 ADDRESS-LISTS.DC
  15 SAUCERS.BS                    9 ASTEROIDS.BS
  25 LUNA-LANDER.BS               27 AdrList.BS
   6 Month.BS                      6 TICKLER.BS
   3 MORTGAGE-COMPARE.BS           2 ROSES.BS
  10 SNOOPY.BS                     2 POLYGRAPH.BS
   3 FLASH.BS

Disk PGL-V-11 has 16 files on it, 349 sectors in use.
  12 ccglobals.CS  21 CCIO.AS     A SMALL-C compiler.
  28 cc1.CS        28 cc2.CS      17 cc3.CS     12 cc5.CS
  12 cc6.CS        12 cc7.CS      22 cc8.CS     14 cc4.CS
   5 ccio.CS       63 CCLIB.AS     1 SETUP.AS   81 cc.GO
  12 Introduction.DC

Disk PGL-V-12 has 6 files on it, 275 sectors in use.
 102 TERMINAL-1p2.AS   126 ADB-4p3.AS   10 TERM.GO
  14 ADB.GO              1 TERM.DC      18 ADB.DC

Disk PGL-V-13 has 7 files on it, 274 sectors in use.
  40 SYMBOL.TX     87 SYSTEMIO.TX   19 CLOCK.TX
  75 FUNCTIONS.TX  14 STORAGE.TX    34 PARSETABLE.TX
   1 8048_Cross_Assembler.TX

Disk PGL-V-14 has 11 files on it, 290 sectors in use.
  87 EXPRESSION.TX   25 8048-TEST.TX   11 MAIN.TX
  30 PSEUDO.TX       63 PARSE.TX       16 CASM-DOC.TX
   4 TODO.TX          2 ASMC.TX         8 FRONT.TX
  39 casm.GO          1 8048_Cross_Assembler.TX

Disk PGL-V-15 has 11 files on it, 348 sectors in use.
  40 SYMBOL.TX      77 FUNCTIONS.TX   87 SYSTEMIO.TX
   9 MAIN.TX         7 FRONT.TX       14 STORAGE.TX
  37 PARSE-FUNC.TX  41 PARSETABLE.TX  29 PSEUDO.TX
   2 ASMC.TX         1 Z-80_Cross_Assembler.TX

Disk PGL-V-16 has 7 files on it, 327 sectors in use.
  86 EXPRESSION.TX   19 CLOCK.TX   112 PARSE.TX
  44 casm.GO         45 CASMZ.GO    16 CASM-DOC.TX
   1 Z-80_Cross_Assembler.TX

Disk PGL-V-17 has 19 files on it, 330 sectors in use.
   8 FILECREATE.BS      8 INITIAL.BS    27 ROOT.BS
  21 POST.BS            7 ENTER.BS      10 STATUS.BS
  11 CREATE-INDEX.BS    1 DAT.DT        17 DESCRIPTION.TX
   1 HEADER.TX         22 CHAPTER1.TX   46 CHAPTER2.TX
  12 CHAPTER3.TX       62 CHAPTER4.TX   57 INTRODUCTION.TX
  10 TITLE.TX           2 SETUP.TX       1 PRINT-MANUAL.TX

Disk PGL-V-18 has 47 files on it, 308 sectors in use.
   3 Copysys.TX        1 Unsys.GO      14 FILMS.GO
   2 Graf.OV           1 I/O.BS         5 PLOTTER.BS
   9 Basic.SY          1 Sexy.OV        9 FRED.BS
   3 TEXT.DT          15 EDIT.TX        1 STAN.BS
   1 DRAW.BS           2 SQUARE.BS      4 SHATAS.BS
   3 POLY.BS           3 8813.BS       11 Initial.BS
  13 Printer.GO        4 POLYDEMO.BS    3 STUFF.BS
  11 fred.BS           1 i/o.BS         6 plotter.BS
   3 poly.BS           4 8813.TX        9 INITIAL.BS

Disk PGL-V-19 has 17 files on it, 315 sectors in use.
 260 ROSE-DATA.DT      1 PLOT-DATA.DT   3 TEXT.DT
   1 Sexy.OV           2 How-to-use-the-Demo-program.TX
   2 INITIAL.TX        9 INITIAL.BS     1 GARBAGE.DT
   4 TEXT.DT           3 CTEXT.DT

Disk PGL-V-20 has 24 files on it, 350 sectors in use.
  58 IMPS52.BS        58 IMPS52/1.BS   62 IMPS52/2.BS
  28 IMPS52/3.BS      14 STEROT6.BS     2 CENTIGRADE.BS
   5 KELEK.BS          7 EH-PH.BS      12 MOLWT.BS
   8 SPECT1.BS         2 ARRHENIUS.BS   4 XLENGTH.BS
   5 POLAROGRAPHY.BS   4 Eyring.BS      3 Arrhenius.BS
   3 GUGGENHEIM.BS    25 LLSQPLOT.BS    7 PKA.BS
   9 CVK.BS            3 KINCHANGE.BS   2 DISTANCE.BS
   3 RCALK.BS         11 SPECTRAL-ANALYSIS.BS

Disk PGL-V-21 has 10 files on it, 346 sectors in use.
   0 WINDOWS FOR THE POLY!            8 HELLO.TX
 111 DOCUMENTATION.TX   75 DEMO.BS    2 BSdef.ED
  50 General-Programming-Functions.BS  9 PRIME.BS
  10 INTRO.BS           23 INSTRUCTIONS.BS
  52 SUPERMAN.BS

Disk PGL-V-22 has 8 files on it, 350 sectors in use.
 131 YPS-42.BS        40 AS-DATE.BS    65 CREATE/BANK.BS
  24 AS-DATE-OMNI.BS  60 CASHFLOW.BS
  13 MORT2.BS          9 FIFO-LIFO-INVENTORY-COMPARISON.BS

Disk PGL-V-23 has 17 files on it, 342 sectors in use.
  22 OTHELLO.BS      1 STARS.DC      27 STARS.BS
   9 ACEY-DEUCY.BS  11 CANNON.BS     14 CHASE.BS
   3 RUSSIAN-ROULETTE.BS             67 SUNWAR.BS
  14 TIMEBOMB.BS     9 TANKS.BS      16 SLOTS.BS
  66 OREGON.BS       9 BAGELS.BS      9 AUTORACE.BS
  12 POUNCE.BS      32 REBEL.BS      17 BOXING.BS

Disk PGL-V-24 has 27 files on it, 338 sectors in use.
  31 MDIR.BS                  3 MDIR.DC
  27 RPN-CALCULATOR.BS       11 RPN-CALCULATOR.DC
  32 PAYROLL.BS              12 LINEAR.BS
  10 LREGRS.BS                8 PEEKER.BS
   6 T-TEST.BS                3 OBV-ADDER.BS
   6 OBV/VPT/MA-CORRECTOR.BS  4 HOURS/DEMO.BS
   8 8THS/MA-CORRECTOR.BS     5 REMS/LINES/OUT.BS
  24 TELED-11-13.DT          45 STOCKS49.BS
   5 STOCKS49.DC             36 GOLD6.BS
```

```
  3 GTEST.DT                  1 GOLDSPOTS.TX
  2 DISK.TX                   9 COMPRESS.BS
 19 EXPAND.BS                 5 COMP-EXP-EXAMPLE.TX
  6 FRACTION-METRIC-CONV.BS   6 COMPARE/TWO.BS
  7 BASE.BS

Disk PGL-V-25 has 48 files on it, 348 sectors in use.
  7 BREAKOUT.GO   10 Chase.GO     2 NULIFE.DC
  3 NULIFE.GO      2 TWEAK.DC     1 TWEAK.GO
  2 SYSMOV.DC      1 SYSMOV.GO    3 SEARCH.DC
  2 SEARCH.GO      2 XDUMP.GO     2 compare.GO
  7 FORMAT.DC     16 FORMAT.GO    1 compare.DC
  8 TBASIC.GO      5 Tbfn.OV      2 XDUMP.DC
 10 Gnomus.GO     57 Gnomus.AS   10 Gnomus.DC
  1 BIT.GO         1 SCREEN.GO    2 ARISE.GO
  2 DLIST.GO      15 FILMS.GO     2 CLEAN.GO
  8 CHECKSUM.GO    8 SZAP.GO      4 DUMP.GO
  1 USING-DUMP.TX  3 WORMS.GO     9 CONVERTER.BS
  7 CALENDAR.BS    5 COMPARE.GO   3 SORT-ROUTINE.BS
  8 COPYALL.BS     5 COMPARE.GO   4 DIRECTORY.BS
  1 Eater.GO       1 Screen.GO    1 ScreenOff.GO
  1 ClockOff.GO    2 Clock.GO     1 KillEater.GO
  5 Screen.DC      3 Clock.DC     2 ASROM-DEMO.GO
 88 MIRROR-MAZE.BS 3 READ-THIS.TX

Disk PGL-V-26 has 29 files on it, 338 sectors in use.
 20 HAMURABI.BS   24 CRAPS.BS    16 HANGMAN.BS
 13 REVERSE.BS    14 HORSE.BS    22 BACKGAMMON.BS
 20 STOCK.BS       7 SHOOT.BS     9 MATHDRILL.BS
 19 PLOT-DEMO.BS   1 PLOT.BS    22 STARTREK.BS
 21 LANDER.BS

Disk PGL-V-27 has 37 files on it, 348 sectors in use.
 12 EXAMPLE.AD     1 ADdef.ED     5 L1CODE.SY   9 REF-CARD.DC
  3 HELP.DC        1 DEMO.CD     95 DOCUMENT.DC 2 STUFF.CD
  9 ANNOUNCE.DC 3 L1CODE-INFO.DC   (LITTLE-ADA COMPILER)

Disk PGL-V-28 has 32 files on it, 350 sectors in use.
 43 GRAPHICS.BS    1 GRAPHICS.DC    2 BIGPRINT.DC
 12 BIGPRINT.BS    8 BIGPRINT.DT    2 DEMO.DC
  6 DRAWLINE.BS    3 FLAG.BS        4 CENSOR.BS
  6 XMAS-TREE.BS  23 HEADER.BS      9 CASTLE.BS
 28 ANSWERS.DT    19 SCROLLING.BS  14 NUMBERS.BS
 23 STATES.BS      5 THE-FLY.BS    24 STATES_&_CAPITALS.BS
  4 HILO.BS        1 DATE.DT        1 DECIMAL-MATH.DC
  4 FLAG/1.BS     17 QUIZ.BS        4 NEAT_MULTIPLIER.BS
  6 DEMO.DT        6 DEMO.BS       24 SPELLING-BEE.BS
  7 MYSTERY.BS     5 LESSON.BS     32 DECIMAL-MATH.BS
  1 SPELLING-BEE.DC

Disk PGL-V-29 has 44 files on it, 348 sectors in use.
  4 SYSTEM.SY    3 Fmsg.OV   23 FORTH.GO    1 TEST.FX
  1 GO.TX        8 Fmsg.AS    5 FN.FX       4 8-QUEENS.FX
  5 VLIST.TX     2 KEY.FX     1 DEMO.TX     1 FIG-FORTH.TX
  7 DOC.TX       5 prin-for-msg.AS      2 prin-for-msg.GO
```


Readers Requests

Readers have asked for articles about the following: Assembly language article governing the use of WH0, WH1, Ckdr, Msg, etc. How would CP/M be of use. How does CP/M work. Where to get Drive Service, Keyboard Service, etc. Also wanted are hardware update recommendations, source lists, Communication software articles, File transfer to other computers. More on PClones. More articles on Hardware (Boards, etc.) More articles on languages. An explanation of relocatable files.

As time and space permits, I will try to answer all these questions. However, our readers are encouraged to submit their own articles on these and any subjects. Articles should be submitted on Poly 5" SSSD disk. PC disk format is also acceptable.

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.) I have just added all the Abstract Systems BugNote files to the help system. So, here are two BUG help files.

```
$HELP BUGS
      HELP file for BUGS.                    10/01/87

Help is available for the following BUGS:

  ASIN  Asmb  Auth  BASIC  boot-1  boot-2  C01L  COMP-DISK
  COPY  DD-FDC  DELETE  DIGITS  Dio-MS  Dio-SD  DLIST  EDIT-1
  EDIT-2  EDIT-3  EDIT-4  EDIT-5  EDIT-6  EXP  FORMAT  FTP
  Gfid-1  Gfid-2  Index  LEN  LIST-1  LIST-2  MS-RAM  OUT
  OVERFLOW  PACK-1  PACK-2  RDB  RENAME  RENAME-1  Setup
  SIN/COS  Sio-1  Sio-2  Sniff  Tran  TYPE/PRINT

Syntax: "HELP BUG [name]"          Example: "HELP BUG EDIT-2"
"HELP BUG [name]" displays the help file for 'name'.
"HELP BUGS" displays this message again.

$HELP BUG Index
      Abstract Systems BugNote Index April 20, 1989

File       Number   Date     Description
DD-FDC        0.0 09/08/82 DD FDC timing tolerances too fine.
C01L          0.0 11/08/82 BASIC C01L requires 00 byte at 0037H.
BASIC         1.1 11/03/82 BASIC C03 Skips 00 bytes on INPUT.
BASIC         1.1 11/03/82 C03 Writes past end of file in INOUT!
RENAME-1      2.0 11/03/83 Wild card .* undeletes and renames.
TYPE/PRINT    3.0 11/03/83 TYPE or PRINT may report two errors.
Dio-SD        4.1 11/03/83 Multi sector read error not reported!
PACK-1        5.0 11/06/83 PACK only looks at first letter.
LIST-1        6.0 11/08/83 # must be defined with a drive number.
MS-RAM        7.0 11/09/83 RAM not disabled during IN or OUT inst.
LEN           8.0 11/12/83 BASIC LEN only accepts whole variable.
PACK-2        9.0 11/30/83 PACK error doesn't Abort Command files.
EDIT-1       10.0 12/01/83 Edit hangs up on a full directory.
RDB          11.0 12/21/83 RDB doesn't restore Front Panel Vector.
boot-1       12.0 12/26/83 boot does not check for a system disk.
ASIN         13.0 01/18/83 BASIC ASIN(-1) gives the wrong sign!
Sio-1        14.0 01/24/83 Kill routine doesn't empty Tx buffer.
SIN/COS      15.0 01/24/83 BASIC SIN & COS rounding error.
EDIT-2       16.0 04/05/83 thinks more to read on exact sector.
```

Sio-2	17.0	04/11/83	Sio dumps incoming characters.
FTP	18.0	04/28/83	Recieve file already exists error.
EDIT-3	19.0	05/05/83	Edit hangs on closed output file.
Gfid-1	20.0	05/10/83	$ can be defined as $ and locks system.
Gfid-2	21.0	05/10/83	Disk Full error may mean file exists.
Dio-MS	22.0	05/12/83	MS Dio DONT set 0 during critical move.
EDIT-4	23.0	05/13/83	CTRL character Global search problem.
OUT	24.0	05/26/83	OUT does NOT restore WH5 after use.
Asmb	25.0	06/07/83	Asmb wants RST R instead of RST N.
Sniff	26.0	06/07/83	accepts directory, reports 0101 error.
boot-1	27.0	06/08/83	accepts directory, locks in error loop.
Setup	28.0	06/15/83	DELETE doesn't reset default printer.
LIST-2	29.0	06/21/83	LIST doesn't set PVEC internally.
FORMAT	30.0	07/05/83	(sind) doesn't change (wund) default.
DELETE	31.0	07/07/83	DELETE x,y.* acts like DELETE x.*.
Tran	32.0	07/16/83	Tran.OV works only for DIGITS 8 or 9.
DIGITS	33.0	07/18/83	DIGITS N works like DIGITS 2*INT(N/2).
COMP-DISK	34.0	07/18/83	COMP-DISK.GO writes past MEMTOP!
Auth	35.0	08/02/83	Auth asks for password on bad name.
DLIST	36.0	09/08/83	DLIST duplicates the last file on page.
RENAME	37.0	01/14/85	REN <d<A <d<sub<B does not give error.
OVERFLOW	38.0	02/19/85	Overflow error on BASIC TRIG functions.
EDIT-5	39.1	04/19/85	ESC library loads on full memory text.
EDIT-6	40.1	05/19/85	ESC ? scrolls long lines off screen.
COPY	41.1	06/30/85	COPY name-with-*.EX locks up system.
EXP	42.1	01/07/86	Overflow error given by BASIC EXP.

If anyone wants more detailed information on any of these bugs, let me know, and I'll print the entire bugnote in a future issue of PL. If you're in a hurry, call. Also, if you know of system bugs not listed here, let me know.

In This Issue

Coming Soon

Modems and Communications software; More: BASIC for Beginners, System Programmers Notes, Help, BugNotes, Public Domain Software, etc.

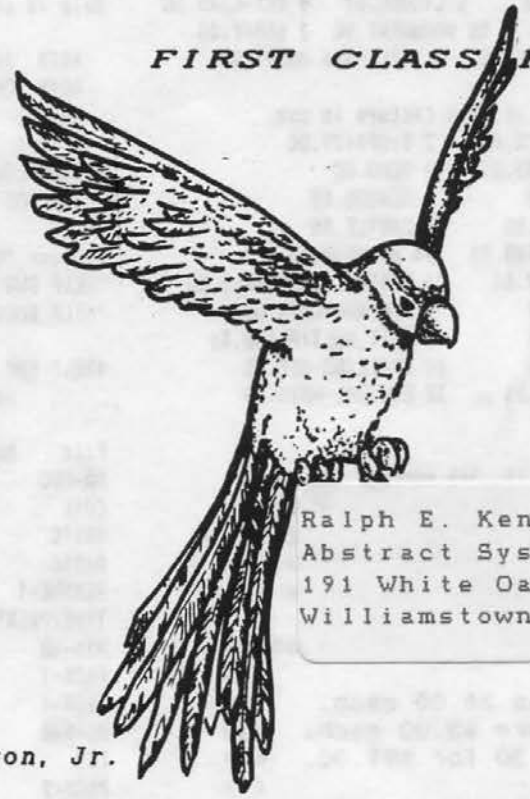
Questions

Can you find and answer the questions asked in this issue? Send your answers and requests in.

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

Address Correction Requested

FIRST CLASS MAIL



Ralph E. Kenyon, Jr. EXP: 9999
Abstract Systems, etc. 184
191 White Oaks Road
Williamstown, MA 01267-2256

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 ads are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* (1980 thru 1988) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 - $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 ($3.50, $4.00, $5.00).

PolyLetter

Editorial

(This issue of PolyLetter is delayed by downtime; see my comments elsewhere.)

Last time I reported that I had purchased used disks from Poly. Among those disks I have found a partial implementation of UCSD Pascal. Of course, it was ROM dependent and didn't work. But, with the help of my trusty DisAssembler, I have managed to find out why not and patch it to work. It's an interesting system, though incomplete, which has several utilities and is essentially a development system. Since then I have found out that UCSD Pascal has been sold to Pecan Software. I contacted them about the possibility of releasing this particular (old) version to our users group, and they have declined to make it available as a public domain item, but are willing to lisence it for a nominal fee subject to negotiation. If anyone is interested in this system, I will pursue the matter with more vigor.

My favorite language is ADA, with is Pascal based. That means that it is somewhat like Pascal, but a Superset. For those of you who are using PC's, R. & R. Software has a fully validated ADA compiler for the PC called Janus Ada. -- ADA is trademarked by the U.S. DoD. More on ADA in future issues.

Letters

Ralph, May 16, 1989

Good to see PolyLetter going strong in spite of PolyMorphic Systems demise. The Poly Meta compiler and Orange compiler look pretty interesting. Will contact you. -- Allen Daubendiek -- Palm Bay, FL

Dear Ralph May 20, 1989

C PROGRAMMING - After the PolyLetter articles from Bybee, you and Cain, I found an excellent tutorial book by Cochan ($24.95) which I expect to enable me to generate some more effective and portable programs. ... The tutorial starts hands on programming exercises in the very first chapter and I need to get going right away. Here's a tip. To avoid EDIT cancellation due to insufficient directory space, create a reserve 4-sector subdirectory on every DD or 8 inch disk called RDIR.DX that can be named to accommodate an EDIT output. -- Earl Gilbreath, Savannah, GA.

[Earl is using Poly's to mechanize the Chatham County Census of 1870. That census is still in the original hand written form and consists of 1039 pages with 40 entries per page. To complete the project, Earl has 30 volunteers ranging in age from 25 to 70, most of whom had never seen computers. Earl says:

"They have loaded the 5-1/4 inch disks into [the] drives in every imagineable way and 3-4 times have jammed them so [the disks] had to be forcibly yanked free. In no case, however, have we lost any data on the disks! I'm astounded. They all seem to enjoy the experience, though."

Earl also asked for an explanation of WAIT states and how come I built the HELP system. Those I answer elsewhere in this issue, along with further comments about the Directory Full situation. -- Ed.]

PolyLetter May 24, 1989

I like my Poly, but as of 3/89 I'm primarily reliant on a 20 MHz 386 AT clone with a Canon BJ-130 printer (good printer! 360 dpi ink jet). PolyLetter would have ongoing value (less esoteric) as a journal with and IBM compatible slant with residual Poly information. It would be more personalized than the monster magazines with large circulations. -- Robert Stricklin, Warrenton Oregon.

Dear PolyLetter May 24, 1989

How about more reader participation! -- Karl Thomas, Elk Grove, IL.

[By all means! -- Ed.]

Announcements

A press kit has been received for BusinessVision II, which is a "Business Management & Accounting System" by BusinessVision Management Systems, Inc.,

Forth Floor, Citicorp Plaza, 8410 West Bryan Mawr Avenue, Chicago, IL, 60631, the U.S. offices of the Ontario Canada firm. BusinessVision II retails for $995, but includes an integrated package with the following services: Accounts Receivable, Accounts Payable, General Ledger, Inventory Control, Order Entry and Billing, Sales Analysis, and Payroll, with Automatic Posting. An impressive demo disk with a 5 minute presentation was included. The president of BusinessVision is Murray Aston. Call (312) 380-1221 for more information.

for Football Fans

A sample copy of the 1989 Pro Football Pointspread Analyzer has been received from Best Bet Software, 3104 Shattuck Avenue, Berkeley, CA 94705, (415) 540-5072. The flyer states: NFL and NCAA football computer databases for checking theories and discovering patterns. Instant access to every NFL game from 1979 to present. Instant access to every game for over 100 colleges since 1984. "the finest handicapping tool of its kind" -- The Gold Sheet Handicapping staff. The software comes with a 35 page manual.

[Didn't someone have something similar for the Poly at one time? -- Ed.]

MicroTac announced the release of "The Language Assistant Series", for the IBM line, which is to help one study and write in a foreign language. The system boasts on line Random House Bilingual dictionaries, complete verb conjugations, grammar help topics, support for accented characters, and that it can work with word processors. Spanish and French available immediately, German and Italian versions scheduled for release in September 1989 -- $49.95 each. Contact MicroTac, 4655 Cass Street, Suite 304, San Diego, CA 92109, (619) 272-5700 for more information.

DownTime

Unfortunately, this issue of the PolyLetter is delayed by some downtime. We like to believe the Poly doesn't have downtime, but that's more wish that reality. My experience is that the Poly is very reliable, but not 100%. But, on with the story.

I have a clock calendar board in my system which has a nickle-cadmium battery to keep the clock running when the Poly is turned off. Recently, I began to see some very strange things happening. The Poly would simply stop. I'd be editing and the cursor would stop moving. Control-Y, and Control-Z did nothing. It was locked up tighter than a drum. Well, There was nothing I could do except re-boot. Things seemed to go fine for a while, but then there it was again -- dead. After twice in one morning, I was afraid to pack the disk. What if it died in the middle of the job?

The way PACK works is to read all directories, compute the new locations for files that have to be moved, update the directories, rewrite the directories, and finally move the files themselves. Imagine the horror story if the directories had been re-written, but the files hadn't been moved and the system died.

On a 350 sector disk with only a few files, the recovery job wouldn't be too bad; but on the hard disk? With lots and lots of files? Forget it! It was too risky to chance. Well, after a couple of more days of work without packing, the disks began to fill up. Also, I had a half dozen more incidents of the system dying.

Well, I got in an order for EPROMS, and decided I could go ahead and burn them without needing to worry about packing the disk. Well, I got about half through the job and the system stoped buring the proms. I'd get a verify error, and using Szap to check, see that the EPROM hadn't been programed. Well, maybe both these problems are connected. After all, I have had problems in the past with corrosion on the chips and cleaning the chips and the boards seemed to help. So, I decided to open the system up and clean it up.

After taking apart the Clone that sits atop my Poly, and removing the connections and cover, I lugged the parrot out to my picnic table, where I had my vacuum set up. When I started removing the boards, I discovered that the nickle-cadmium battery had leaked. The corrosive acid ran down the board and into the S-100 bus connector corroding about 12-14 of the connections in the connector and on the board itself. Talk about a mess! Ah ha! Here's the problem, sez me. Well, I did vacuum and blow out the entire chassis, and clean up the board, but cleaning the S-100 connector posed a challenge. I had used emery cloth on the board itself, with liberal amounts of contact cleaner. So, I folded a piece of emery cloth over a thin flat piece of cardboard and pushed it in and out of the S-100 connector several times. I also used liberal amounts of contact cleaner, and vacuumed the remains up with a crevass tool. There is still discoloration on the connectors, and I may need to replace it in the future, but, for now, it seems to work. The EPROM burner card still didn't work, but that's another story for another time.

Please share with us your own experiences with downtime or other hardware difficulties. Please describe the symptoms which made you think you had a problem,

what you did to find out the cause of the problem, what you evaluated the final cause to be, and how you fixed it. Now that Poly is closed, and there aren't too many places which can service Poly's, we need to share our experiences in this area to help everyone keep their Poly's running. So, send in your experiences.

Wait States

Earl Gilbreath asked for an explanation of "wait states". The explanation starts with some background on how computers operate. The central processer unit (CPU) operates on a "clock" signal which is constantly "ticking" between logical states "0" and "1". Logical states are instantiated in these physical devices as two voltage ranges. In positive logic, a "0" is represented by a voltage less than .5 volts, while a logic "1" is represented by a voltage greater than 2.4 volts (usually less than 5 volts.) So the clock signal is a line in which the voltage switches back and forth, usually from about .3 volts to 3 volts.

One clock cycle is the time it takes for the clock signal to go from 1 to 0 and back to 1. In the Poly that time is 543.534 nano-seconds, or about 1/2 a micro-second. The Poly clock speed is 1.8432 MegaHertz (MHz).

The simple explanation of a wait state is that some devices can't run as fast as the CPU, so it sends a signal to the CPU to WAIT one clock cycle more than normal. The net result is that the system runs slower whenever that device is being accessed. In the IBM, with its 5.77 MHz clock speed, one wait state would be 173.3 nano-seconds long. Since a machine cycle is 4 clock units in the IBM, one wait state would be about a 25% slow-down in the CPU, but this would only be noticable on tasks which spent a lot of time accessing the slow device.

HELP — How come?

Earl also asked for an explanation of why I built the HELP system. Well, when I first began to look at Poly's earliest Exec -- Exec/4D, I found that it had a built in HELP command. It would look for and type a file called Ht.Ht to the screen whenever one typed "HELP". Poly removed this feature from later Exec's. To me, it seemed a shame that this feature was discarded.

I began to keep a disk of system info files and would put it in to see things I had forgotten. Eventually, I started putting documentation files in for various programs. When I got my IBM compatible, I got a disk with a help program on it. One typed HELP and a topic name, and the program displayed the text on the screen. I decided it was about time I had the same feature on the Poly and wrote HELP.GO as a program to do the job. It just looks up files with a default extension of ".HF" and in the subdirectory "<?<HF.DX".

As I began modifying Exec and correcting bugs, I also started adding features. Naturally, I added HELP as a system command for Exec/(A|S). When I was programming at college on the DEC system 10, I found it had a nearly complete help system. Once I made HELP part of Exec, I decided that I ought to have HELP files for everything one could do on the Poly. I began to write HELP files for all the system commands and for many of the programs. Sometimes it was to help me remember things, and sometimes it was to make it easy for others to use my system. Eventually, the system just grew and grew and now requires 4 SSSD diskettes. I've even begun adding files to help me remember how to get PolyLetter out. With press time coming only once every two months, I forget how to do renewal and reminder notices, so have put that stuff in custom HELP files. You can add help files to the system for anyone using your computers. Have the INITIAL file end with the HELP command, but custom tailor the HELP.HF file for the application.

Disk Directory full!

Earl suggested putting a sub-directory on every disk so that when one got the "Directory Full" message from edit, one would have a place to put the file. Well, the operating system does allow for this condition, but it is undocumented.

When you get EDIT's DIRECTORY FULL notice -- even if there is not room for another file, there is enough room to create another directory. So, even if you don't have the any sub-directories, you can create one, because Gfid reserves space for just such an emergency. It would have been nice of Poly to tell us this.

But, Earl's suggestion is a good one. I have a spare directory on nearly every disk. To speed things up, the spare directory should be put on before anything else. When Gfid is reading subdirectories it has to go get the directory, and if the directory is at the end of the disk you'll have to wait while Dio steps out to the end of the disk and then back to the beginning.

DIGITS

Earl Gilbreath of Savannah, GA reported that there is little information on BASIC "DIGITS", and that he found the DIGITS declaration could not follow DIM's in BASIC C04. Here is some additional info.

BASIC keeps a flag somewhere that tells it how many digits to print in BASIC. The default is 8, but there is a problem because BASIC really gives 8 digits of accuracy when 9 are specified. One should always ADD 1 to the number of digits of accuracy one actually wants to be sure BASIC will print enough digits.

```
DIGITS 8        DIGITS 9        DIGITS 10
PRINT 1/3       PRINT 1/3       PRINT 1/3
gives           gives           gives
 .33333333       .33333333       .3333333333
```

When numbers are stored in binary coded decimal (BCD) format, each nybble represents a single digit. One byte can store one or two digits; two bytes can store three or four digits. My guess is that BASIC takes the number specified by DIGITS, divides it by 2 and then stores that as the number of BYTES of storage for the mantissa of a number. What one actually gets is (in BASIC)

```
DIGITS 2*(INT(N/2))
```

(See [A:S] BugNote 33 = HELP BUG DIGITS)

However, whenever a DIM statement is executed, or a reference is made to a variable which has not been previously used, BASIC reserves space for the variables. The DIGITS command may change how much space is needed, and BASIC doesn't know how to change the space allocated. I first discovered the problem a while ago, but didn't write it up as a BugNote. The Poly BASIC Reference Manual states on Page 8:

"When DIGITS is used, all variables are CLEARed. Therefore, the DIGITS statement should normally be the first statement in a program."

Poly says the DIGITS statement should be the first one in a program, but it really needs only to be used before any storage is referenced, because it automatically cleans the storage slate just like the CLEAR command. A little experimenting shows that the CLEAR command does NOT reset DIGITS back to 8. So, if you have one of these programs which needs to set DIGITS to a default value, and then ask the operator if he or she wants to change it, the program should probably have both a DIGITS 8 statement and a DIGITS N statement.

```
10 DIGITS 12 \REM Default value for this program
20 INPUT "Please specify DIGITS needed (6-26)",N
30 IF N<6 or N>26 THEN 20 \REM Out of bounds trap
40 N=2*INT((N+1)/2) \REM Bug correction - Add 1 to ODD values
50 DIGITS N \REM Make it happen (Also, N will become 0 now).
60 REM Somewhere later is a "GOTO 10" statement.
```

This raises a problem with how to get BASIC to remember just what value of DIGITS the user specified. One way is to write it to a file and then read it back. Another way is to POKE it into some non-USER memory location and then PEEK it back. A third way is to put it in the keyboard buffer and then read it back later, but making this work if the program is being run under command file mode is a bigger problem. Here's a demo I came up with to test the problem.

```
BASIC
System 88 BASIC C04, 10/28/81.  33479 bytes free.
>110 DIGITS 10 \REM Set program default digits value.
>120 DIM A$(1:64) \REM Temporary storage space keyboard buffer
>130 INPUT "Digits ?:",N \REM get actual DIGITS from user.
>140 A$=STR$(N)+CHR$(13)+STR$(PEEK(11656),%$11)+CHR$(13)
>150 A$=MID$(A$,2,63) \REM drop leading blank
>160 POKE 11656,0 \REM Turn COMMAND file mode off
>170 REM Now we get what's already in the Keyboard buffer
>180 IF INP(0)<>0 THEN A$=A$+CHR$(INP(1)) \GOTO 180
>190 OUT(0),A$ \REM DIGITS, CMDF, & previous stuff to buffer
>200 DIGITS 2*INT((N+1)/2) \REM Correct for odd DIGITS bug
>210 PRINT "Getting digits ", \INPUT1 "",N
>220 PRINT " back from the keyboard buffer."
>230 PRINT "Getting CMDF ", \INPUT1 "",K
>240 PRINT " back from the keyboard buffer."
>250 POKE 11656,K \REM Restore COMMAND file mode
>260 PRINT "Digits specified by User is:",N
>270 PRINT "Command file mode is ",
>280 IF K=0 THEN PRINT "off." ELSE PRINT "on."
>290 REM Note that when we get here A$ is not defined
>300 DIM A$(1:20) \REM just to show it works
>RUN
Digits ?:9 <-- Input value of digits from COMMAND file
Getting digits 9 back from the keyboard buffer.
Getting CMDF 1 back from the keyboard buffer.
Digits specified by User is: 9
Command file mode is on.
>REM and now Input value of digits from operator
>RUN
Digits ?:13
Getting digits 13 back from the keyboard buffer.
Getting CMDF 0 back from the keyboard buffer.
Digits specified by User is: 13
Command file mode is off.
>BYE
(Exec/[A:S] 20-DEC-87)
```

One thing this routine does is preserve any type-ahead when it uses the keyboard buffer for temporary storage. We Poly users often input one or two more commands to the keyboard buffer while a command file is still running. Our temporary use of the keyboard buffer to store the command file mode flag and digits value uses 4 or 5 characters, and it is rare that anyone types more than 59 characters ahead.

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 BASIC DIGITS
```

HELP file for BASIC CONTROL COMMAND "DIGITS".

"DIGITS <n>" specifies the precision for numerical variables. n may be from 6 to 26. 8 is standard.

In a program the command DIGITS n must be specified prior to any reference to any numerical variable. This is because the space allocated for a numerical variable is changed. The variable storage area is cleared when DIGITS is executed.

HELP BUG DIGITS

Abstract Systems BugNote 033.0 July 18, 1983

DIGITS

BASIC C03, 04/14/81 has a bug in the DIGITS keyword. Digits is supposed to be able to set the number of digits to be used in calculations from 6 to 26. DIGITS actually only sets to even numbers. DIGITS 7 gives the same results as DIGITS 6 DIGITS 11 gives the same number as digits 10, etc. Demo:

```
$BASIC
System 80 BASIC C03, 04/14/81.  33355 bytes free.
>100 DIGITS 6
>110 PRINT %21,6,\PRINT 1/7
>120 DIGITS 7
>130 PRINT %21,7,\PRINT 1/7
>220 DIGITS 12
>230 PRINT %21,12,\PRINT 1/7
>240 DIGITS 13
>250 PRINT %21,13,\PRINT 1/7
>260 DIGITS 14
>270 PRINT %21,14,\PRINT 1/7
>280 DIGITS 15
>290 PRINT %21,15,\PRINT 1/7
>300 DIGITS 16
>310 PRINT %21,16,\PRINT 1/7
>320 DIGITS 17
>330 PRINT %21,17,\PRINT 1/7
>460 DIGITS 24
>470 PRINT %21,24,\PRINT 1/7
>480 DIGITS 25
>490 PRINT %21,25,\PRINT 1/7
>500 DIGITS 26
>510 PRINT %21,26,\PRINT 1/7
>RUN
 6 .142857
 7 .142857                    <= Same as DIGITS 6
12 .142857142857
13 .142857142857              <= Same as DIGITS 12
14 .14285714285714
15 .14285714285714            <= Same as DIGITS 14
16 .1428571428571429
17 .1428571428571429          <= Same as DIGITS 17
24 .142857142857142857142857
25 .142857142857142857142857  <= Same as DIGITS 24
26 .14285714285714285714285714
>BYE
```

To avoid this bug, set DIGITS at least one more than you need, and use print format statements to set the number of digits to be printed.

Modems

Well, I have been promising an article on modems and modem software for some time now. Well, Here it is, such as it is.

BASICS. 'Modem' stands for 'modulate demodulate', and comes from the earliest technology for encoding information. Modulation is a recent method of encoding information.

A long time ago (in a distant galaxy) the telegraph was invented to send signals over long distances. Samuel F. B. Morse is generally credited with inventing the telegraph in 1837, but he was building on the work of earlier inventors. Morse gave us a workable system along with a (now standard) method for encoding information. Nearly everyone knows that the internation distress call 'S-O-S' (Save Our Souls) is encoded in Morse code as dot dot dot, dash dash dash, dot dot dot. This signal can be sent as a series of sounds (on a foghorn or similar device) a series of blinking lights, or as a series of electric pulses (which usually get transformed into light or sound).

In the Morse telegraph one operator keys an electric circuit which causes a sounder to respond at other stations on the line. Letters are sent as a series of intermixed "dots" and "dashes" or "short" and "long" pulses. Experienced telegraph operators can send and receive very fast so their idea of "long" can sometimes be shorter than an inexperienced operator's idea of "short". (The earliest baud rate mismatch problem.)

From the information perspective, "shorts" and "longs" are the only two possibilities, so can be represented as 0's and 1's. Early telegraphs depended upon electric currents being started and stopped to operate relay's which use a lot of electric power. In an attempt to reduce the cost of sending messages, designs were thought up to send more than one message at one time. "Multiplexing", as this is called, used a carrier tone which was turned on and off in amplitude modulation, and shifted to another tone and back in frequency modulation. In multiplexed systems, more than one set of tones is included. Duplex modems have send and receive tones allowing sending and receiving at the same time. Because the implementation of frequency modulation in data communications sent only "shorts" and "longs", (0's and 1's), the frequency modulation used just shifted between two frequencies whenever the key was touched, and is called "Frequency Shift Keying" (FSK).

The earliest cassette based data storage devices used this technique to record data on an audio tape. The Kansas City Byte standard encoding method used two tones, 300 Hz and 600 Hz. The cassette interface card produced a signal suitable for an audio recorder input consisting of asynchronous ASCII encoded characters as a stream of tones.

ASCII? Asynchronous? Experienced Morse telegraph operators have no trouble perceiving just which short and longs form which letter. One of the techniques was to allow just a little bit more time between letters than between individual shorts and longs within a letter. Also, the telegraph line could be idle for long periods of time. Usually the operator would send an "attention" signal before the start of a message. Often the receiving operator would send an "acknowledge" signal before the sending operator started the actual message. This process could be called "establishing synchronization". Once the sending and receiving operator are sure each other is listening, the sending operator can send the characters fast and furious with hardly any space between characters.

This is the essential difference between synchronous and asynchronous data transmission. But, I 'll expand on this a bit later. Meanwhile, let's look at "ASCII". 'ASCII' stands for 'American Standard Code for Information Interchange', and is an "official" list of the shorts and longs used to represent each of 128 characters. Seven bits are used because 2^7 is 127. (A "bit" is a measure of information and corresponds to a single short or long.)

First, let's look at asynchronous data transmission. Asynchronous just means not synchronous -- letters are sent one at a time with no particular timing involved. Like in the teletype case an attention signal is required, so a "start bit" is sent first. Then the data bits are sent (low bit first). Finally one or more "stop bits" are sent. In a modem, this means that the modem idle tone is held until time to send data. Then the frequency is shifted to the lower tone. This signals the receiver to begin checking the signal for data bits. If the tone remains low a one is sensed, if the tone shifts, a zero is sensed. The receiver counts the proper number of bits and then looks for a stop bit (high tone) after the right number of bits have been counted. If the high tone is not seen, it reports a framing error.

If this seems like we haven't been told everything, that's right, because the sender and receiver have to be set up in the same way. How much time is there between bits? How many data bits in a character? Is there a parity bit? If so, is it odd or even parity?

How much time between bits is determined by how many bits we can send each second. One baud is, for all practical purposes, one bit per second. 300 baud is 300 bits per second, and the inter bit time is 1/300 sec or 3.3 ms (that's milliseconds, not Micro seconds). The earliest teletypes transmitted at 110 baud and had an interbit time of 9 ms. Since each character took 7 bits plus one start bit and two stop bits, this meant that each character took about 90 ms. The top speed of these teletypes was about 130 words per minute (5 characters per word). (The hardware was even slower.)

Ok, we have to program our modem by telling it whether it is to be in the send or receive mode (can't both talk at once), what baud rate to use, how many data bits, whether it has to use a parity bit, and how many stop bits to send. Once these are agreed to and set up by both modems, then data communication is possible.

Most of these parameters are set in the Poly by controlling the 8251A USART chip. 'USART' stands for Uniform Synchronous Asynchronous Receiver Transmitter, and is a chip designed to take care of most of the housekeeping chores involved in serial data communications. As the names suggests, it can be programmed for synchronous or asynchronous modes, and can send and receive data. Poly's printer driver Setup program allows us to set the baud rate to any one of 15 different speeds, but the other parameters are "built in" to the software. They are set to 7 data bits, odd parity, 2 stop bits. Since many printer and modems are set to 8 data bits, no parity, this usually does not present a problem except on incoming data. Also, many telecommunications services use 8 data bits, no parity. Poly's FTP sets up the USART for 8 data bits, odd parity, 2 stop bits.

Theoretically, this technology can work up to a very high frequency, so can achieve a very high baud-rate. On the Poly the highest (easily programmable) baud rate is 9600 baud (about 11,000 words per min.) But, there are limitations when it comes to using the telephone lines. The telephone lines were designed to carry voice signals and have a bandwidth of 2600 Hz. Recall I said that duplex modems have send and receive tones? The sender shifts between frequencies of 1270 Hz. and 1070 Hz. The receiver shifts between frequencies of 2225 Hz. and 2025 Hz. This keeps the two channels far enough apart to prevent interference. If the tone were to approach the 2600 baud upper limit, signal loss and

degradation would be enough to cause data loss. Now, technology to distinguish between the two frequencies needs to monitor the signal long enough to tell the two tones apart. Two cycles is about enough time. Two cycles of the lower tone corresponds to about 600 baud, which is the practical upper limit for FSK technology using telephone lines. Of course, early hardware had its own limitations and a factor of two safety margin gives the nice round figure we all remember of 300 baud. Remember when all the modems were 300 baud?

Short haul, or Null modems, which have their own wires, are not limited by the telephone lines. That's why printers can be connected to the Poly at 9600 baud -- no Ma Bell to deal with.

Of course using a modem on the Poly usually meant losing the printer and changing the serial card header plug (unless you were fortunate enough to have two serial cards installed). Connecting the modem in this configuration meant disconnecting the printer at the same time because there was only one 8251A USART, and it could be connected to only one thing at a time.

When I went looking for a MODEM, I wanted to have it and the printer at the same time, so I looked for an internal modem. I got the D.C.Hayes MICROMODEM-100, which went into the Poly backplane and could be run at the same time as the printer. But, I had to write the software to run it. When they first came out they were priced at $400.00.

The first modem I heard of was an acustic cat modem. It had a cradle with a speaker and earphone onto which one placed a standard telephone handset. There were little rubber cups which held the phone in place. Basically, this device had a switch for selecting between originate and answer mode, but left everything else up to the operator, or the computer.

First turn on the computer, and then turn on the modem. Bring up FTP and select the baud rate. Now, Dial the number and listen for the tone. Then plug the phone into the modem and hope the line wasn't too noisy. But when you were the first kid on the block with one, it was terrific, even when it was bad (like sex). These acoustic modems could be had for $150.00 back in 1979, but I saw them at computer shows for $1.00 apiece recently (and they weren't selling)!

One way to eliminate some of the noise is to eliminate the double conversion into sound. Soon modem companies advertised "direct connect" modems. These modems had to have special circuitry built in to satisfy the telephone company and the FCC. The MICROMODEM-100 was one of the first (that I heard about) to offer these features. They hyped their "patented microcoupler" which allowed direct connection.

Modems and cassette storage systems used the same data encoding methods. The next big jump in modem speed was to the 1200 baud modem. This jump involved more precise timing requirements and a "newer" technology. Instead of shifting between tones like the FSK method, the 1200 baud encoding method shifted the phase of the carrier signal itself. Remember Poly phase encoding on the Poly-88 cassette interface? Yep. You guessed it. 1200 baud modems use the same technology. Guess what else? The speed is the same as the Poly phase cassette interface - 1200 baud.

Theoretically, a Poly cassette interface being operated in the Poly phase mode ought to be able to talk to a 1200 baud modem! But, it would be only one way at a time, and the phase shift might not match. I wonder if anyone ever connected two Poly Cassette interface cards together in Poly phase mode? It would be an interesting experiment. Imagine, Poly had 1200 baud modem capability in 1976 and didn't even know it!

I recently obtained a U.S.Robotics S-100 1200 baud modem and tried it out on the Poly. After talking with the company, I found that it could not be used in a fully interrupt driven method. I eventually wrote a routine which used polling to send characters and an interrupt routine to receive characters. That worked ok. The problem was with the design of the modem. They made the modem look like an 8251 from the computer side, but did not include a method of shutting off the transmitter empty interrupt without also disconnecting the modem.

In the Poly's 8251, we can shut the transmitter empty interrupt off and stuff two nulls into its mouth, so we can get back to computing. (If the interrupt is not shut off, the program would loop forever and the computer could do nothing else.)

The U.S.Robotics S-100 modem used the transmitter interrupt enable control to turn the modem's tones off and on. The first time I tried to treat it like the Poly's 8251, it promptly disconnected me. There was no way around the problem. The outgoing characters had to be sent by polling the status and the transmit receiver interrupt had to be physically disconnected for the system to be used.

Once it was working, I found that I

would often get a lot of noise in the form of garbage characters, particularly on long distance lines. I went back to my trusty old 300 baud MICROMODEM-100 and got slow, but reliable data exchanges.

Well, now we are hearing about 2400 and 9600 baud modems. The 2400 baud modems use the same phase shift technique, but with closer timing tolerances and with a smaller amount of phase shifting. Higher than 2400 baud uses the same type of shifting, but increases the speed by going one way at a time, and/or by using some kind of data compression algorithm.

Modem Software

Well, the earliest (Poly) modem software was Poly's File Transfer Terminal (FTP). As I mentioned earlier, FTP uses the Serial port and the on board 8251 USART. It sets up the USART for 8 data bits, odd parity, 2 stop bits. This allows transferring machine language programs between Poly's with great reliability. Any modem which connects to the serial port can be used. FTP itself has only 3 baud rates built in 110, 300, and 9600. But, the correct byte can be changed and set to other baud rates. FTP is also normally set up to use Printer minicard #1, but that byte can be changed also.

In 1980 PolyLetter offered a Modem Kit which included two software programs and a cable. I believe this was actually one of Bob Bybee's first products, before Poly Peripherals was formed. Perhaps Bob will give us the straight scoop.

The Modem Cable was wired so that the user didn't need to open his Poly and change the serial card header.

MODEM-T.GO was a program set up for 300 baud, 7 data bits, one stop bit, and even parity. It could be used to transmit text files, or to save incoming data to text files.

MODEM-T.GO was a program set up for 300 baud, 8 data bits, two stop bit, and no parity. It could be used to transmit data files, or files with machine code in them.

Both programs included the following features, which could be accessed by pressing the ESC key.

The programs presented a menu of options:

B - Begin saving all data into memory
E - End saving data
C - Clear data buffer
S - Save data from buffer into disk file
T - Transmit disk file

Typing any other character (including ESC) would result in that character being sent.

One cosmetic nicity of these programs was the antiword-split feature. Any space arriving in the last 8 screen positions caused the following output to go to the next line. MODEM-T also automatically eliminated linefeeds from incoming files.

Later, in 1983, Bob also offered his Smartmodem package designed to operate the D.C.Hayes Smartmodem 300. The Smartmodem 300 (and later the Smartmodem 1200) both responded to the host computer by allowing many setup commands to be sent to the modem in ASCII.

SM.GO was designed for use with the D. C. Hayes Smartmodem 300. But, because the HAYES command set has become the de facto industry standard for controlling modems, SM.GO will work with many other modems. Bob supplied the source code for SM.GO to buyers to allow users to modify the program easily to fit their own needs.

PGL-V-08 and PGL-V-12 both have assembly language source files for terminal programs.

PGL-V-08 has a terminal program to operate port 0 by Don Moe, another to operate the D.C.Hayes MICROMODEM-100 by Don Hyde of D.C. Hayes Associates, Inc., and another to control the earlier Hayes modem, the 80-103A Modem board.

PGL-V-12 has a program called "Semi-Smart Terminal II", by Stanley Reifel, who used to work at Poly. This program can be operated at 110, 300, 600, 1200, 2400, 4800, or 9600 baud. Here's its main menu:

```
------> Return to terminal mode
        Sends a file
        Save data on the disk
        Begins saving all data into memory
        Ends saving data
        Clears the data memory
        Send out a control Y
        Send out a BREAK
        Set return delay time when
        Set baud rate
        Return to EXEC
```

Finally, there is Abstract Systems HayesSys and FTP-Hayes.GO. FTP-Hayes.GO works just like FTP, but uses the D.C.Hayes MICROMODEM-100 board instead of the serial port. HayesSys, is a system which interfaces the MICROMODEM-100 to the Exec/83 operating system by adding the internal CALL command which automatically uses the MICROMODEM-100 to dial numbers given on the command line. CALL by itself brings up the HayesSys disconnect level operating system allowing setting parameters and various other functions (including a dial command). CALL with a

number automatically dials the number and waits for a modem tone. HayesSys can send and receive text files, or it can log the incoming text directly to the printer.

Actually, I have also written a dial program which only uses the modem to place the call, but not for data connection. This program dials the number on the command line, and then reports "Call completed. Press ESC to disconnect." I pick up the handset, and then press ESC. dial.GO hangs up the modem and then asks me "Did they answer?". If I press 'n', dial gives me back to Exec. But, if I press 'y', then dial issues a command to Exec to edit a file called "CALLS.TX", skip to the bottom of the file, copy the time from the top right corner of the screen (where the clock program keeps it), get the date from the file Date.DT, and add the number just dialed, together with any remarks on the line, to the file CALLS.TX. So, by using the dial.GO program to make all my long-distance calls, I have a complete record of everything but how long the calls were in a data file. I wrote another program to bump this file against the telephone bill, but that has nothing to do with modems. So, I use my modem more as a dialer than for data transfer.

Commercial: I still have a few of these MICROMODEM-100's around if anyone needs one. [See the ads.] I also have one genuine internal Hayes Smartmodem 1200 for the IBM PC and compatibles.

BugNotes

Abstract Systems BugNote 018.0 **April 28, 1983**

FTP

FTP/04 thru FTP/06 have a bug in the receive file routine. When FTP looks up the file name to see if the file already exists, and the file is found, FTP reports error 0258 and aborts the receive file process. A quick calculation shows that 258 hex is 600 decimal. Now, the correct error to be reported when the output file already exists is error 0600. Error 0600 is "That file already exists". It is reasonable to infer that the original assembly source code list was missing the "H" on the appropriate line.

What was:

LXI D,0600

should have been:

LXI D,0600H

This error can be corrected by changing two bytes in FTP.

For FTP/04:
Change byte at 352F from 58 to 00.
Change byte at 3530 from 02 to 06.

For FTP/05:
Change byte at 3529 from 58 to 00.
Change byte at 352A from 02 to 06.

For FTP/06:
Change byte at 3482 from 58 to 00.
Change byte at 3483 from 02 to 06.

Readers Requests

Readers have asked for articles about the following: Assembly language article governing the use of WH0, WH1, Ckdr, Msg, etc. How would CP/M be of use. How does CP/M work. Where to get Drive Service, Keyboard Service, etc. Also wanted are hardware update recommendations, source lists, Communication software articles, File transfer to other computers. More on PClones. More articles on Hardware (Boards, etc.) More articles on languages. An explanation of relocatable files.

As time and space permits, I will try to answer all these questions. However, our readers are encouraged to submit their own articles on these and any subjects. Articles should be submitted on Poly 5" SSSD disk. PC disk format is also acceptable.

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. Noncommercial ads by subscribers are free.

IS YOUR POLY PRINTING TOO SLOW?
>> Quadram Microfazer Printer Buffers <<
64K - Expandable to 2M.
List $299. Special to PolyLetter Subscribers:
Serial to Serial $99. Serial to Parallel $125.
Extra memory available - 256K - $180
Al Levy Associates (516) 293-8368

FOR SALE: 400 4116 memory chips. $200 for the lot. Charles Steinhauser, 404-297-7579

---

1. FOR SALE : Poly 8813 with three drives and MS (DSDD) 8" Disk drive unit mounted in a Poly desk. Monitor and keyboard enclosed in Poly Enclosure. Desk needs refinishing! --- $275 or best offer... I will pay shipping.
2. Poly 8810 with two half height drives, monitor and keyboard. In great shape!

$150 or best offer... I will pay shipping.
3. MS (DSDD) 8" disk drive unit with walnut case. In great shape! $125 or best offer... I will pay shipping.
4. Plenty of used disks to go with the above. FREE to CUSTOMERS of ABOVE.
Bob Kelso -- (318) 981-1971.

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.

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

Address Correction Requested

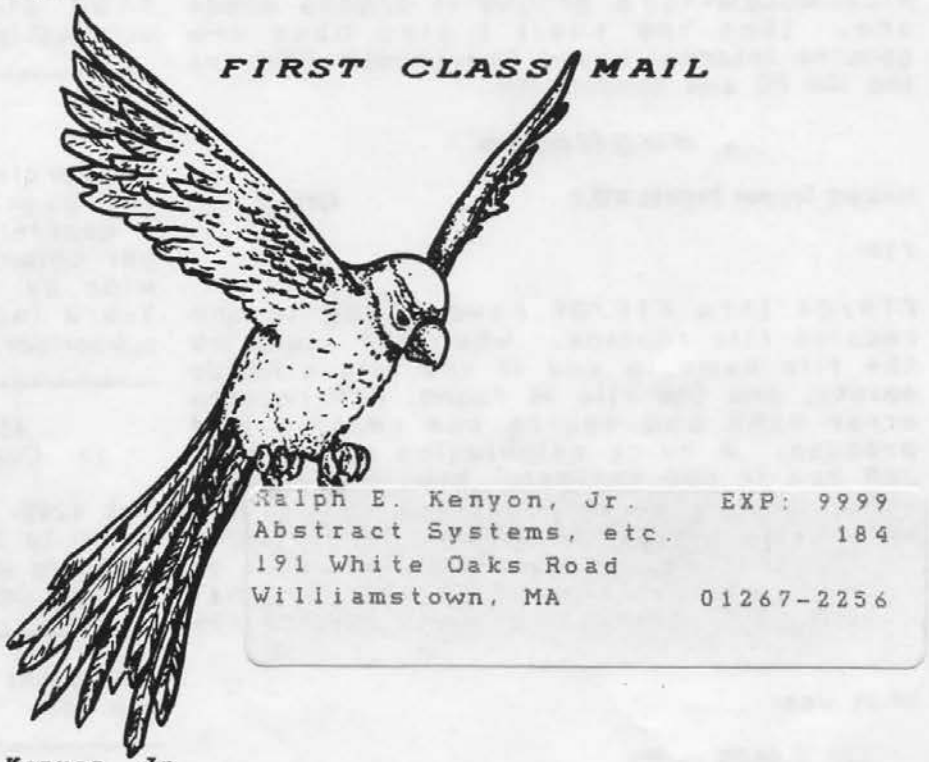
Bit Bucket

What's the newest kind of junk mail? Just the fax please.

In This Issue

Coming Soon

More: BASIC for Beginners, System Programmers Notes, Help, BugNotes, Public Domain Software, etc.



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 ads are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* (1980 thru 1988) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 - $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 ($3.50, $4.00, $5.00).

PolyLetter

Editorial

So far, no one has asked me to pursue the matter of UCSD Pascal. Bob Bybee responded with an article about modems and modem software adding to what I wrote in the last issue. I've also an article about how a new Poly user is putting his Poly to work. But, of significant concern to me these days is environmental issues.

1988 was the warmest year recorded in a century of recorded weather information. 1988 was .34 degrees Centigrade higher that then 1950-1979 30 year average. (1987 was .33 degrees Centigrade higher.) The average temperature of the planet is increasing at an increasing rate, and scientists say that global warming is now dangerously high.

The temperature increase is caused by a build-up of "greenhouse gases", which come from the increased rate at which we consume energy from coal, oil, and wood; together with the fact that we are depleting the world's forests, which would normally remove one of the gases, CO2. The scientists say we have only 20 years left to turn around a century of energy extravagance before the polar ice caps start unreversable melting.

We had almost no snow during the winter of 1988-9 here in New England. The U.S.Midwest drought of 1988 is only a sample of what's coming if we don't take immediate action. I urge PolyLetter readers to take personal action in our fight to preserve our environment by doing two things to start with. Recycle as much as you can and encourage others to do the same. Manufacturing products from recycled materials uses less energy than manufacturing products from raw materials.

Some of us (or our parents) can remember the "war-time" recycling from the WW-II. -- We peeled the labels of our cans, cut the tops and bottoms out, and flattened the sides; we recycled all our metal scraps. I do this now; it only takes a few extra seconds for each can. When my bag of flattened cans is full, I take it down to the metal recycling bin at my local landfill.

Saving our environment will become a full fledged battle and will require war-time conservation and recycling to get the job done. Another reason to recycle is that we are rapidly running out of land fill space.

I urge you all to help save energy by recycling as much as possible. Let it not be said that people into high-tech are the worst offenders.

Letters

Dear Ralph, July 12, 1989
I am glad that you are out there giving us support, condolences and advice. . . . I will be delighted to hear from you as soon as I get current on what Poly is up to. We are a multiple Poly organization and have four that are in use almost daily. Not bad for people with virtually no technical support for the past three years. Poly is such a delight to have and to use. Just today a friend who has a PC-clone was over complaining about not being to diagnose a problem as to whether it was hardware, software, or operator error. I am delighted to have my Poly's and look forward to many years of service still to come. -- Thank you for being there, Richard Wagner, Dallas, TX.

PolyLetter, August 14, 1989
Please list some of the programs available from Abstract Systems in each issue, and how to use system calls (review some of the System Programmer's guide info). -- Ken Lowe, West Valley City, Utah.

[Ken, many Abstract Systems programs have been released to the PolyGlot Library and were listed in recent issues of PolyLetter, but I will list other ones as space permits. -- Ed.]

Announcements

Sanyo has announced price reductions on its turbo PC and AC compatible line of computers ($905 & 1689) for a 1 drive system.

Recreational Mathemagical Software, 129 Carol Drive, Clarks Summit, PA 18411, (717) 586-2784 announced the release of PC-BRIDGE

9.0 which plays the other three hands; and FASTLOAN 3.0 which does loan calculations (what else?). Each retails for $29.95.

Sapiens Software, P. O. Box 3365, Santa Cruz, CA, 95063, (408) 458-1990, is now shipping Star Sapphire Common Lisp 3.0 for the IBM-PC and compatibles. Requires 640K and a hard disk. $99.95.

ETS Center, 849 Seven Gables Circle, Palm Bay, FL 32909, 1-800-833-4777, announced LABLMAKR.PC3 ($59) for making all kinds of labels (including bar-code) and cards for the PC and compatibles.

DownTime-EPROM

Last time I hinted at a problem with my EPROM burner card. My EPROM burner card is a Solid State Music (SSM) 2708/2716 EPROM/PROM card. I recently bought a spare card, but it didn't work, so I had just put it away for spare parts.

My EPROM burner program does more than just program the EPROM. First it checks the EPROM socket to make sure the EPROM is erased. Then it looks up the file of programming data and loads it into memory. Then it tells me to press a key to begin programming and waits for me to press a particular key. Next, it goes though the process of programming the EPROM chip. Finally, it checks the EPROM socket to make sure the EPROM has been successfully programmed and compares that data to the original data. It reports "EPROM Verified." or "EPROM Verify error!" as the final thing it does before giving control back to Exec.

Well, while I was programming EPROMs, the program reported "EPROM Verify error!". When this happens, I usually look at the contents of the EPROM to see what kind of error occurred. Sometimes the EPROM can be erased and reprogrammed again and sometimes it is just plain bad. This time when I looked at the data, it looked like the EPROM had not been programmed at all. Well, occasionally, a bad EPROM won't program at all. So, I tried another one. Same result. Two bad EPROMS in a row? Well, ok, but I'm suspicious of that. The third bad EPROM sent me scurring for my trusty voltmeter to check the EPROG programming voltages. Hmm... 26.3v... close. Well, I tweak it back up to 26.5v and try again. No cigar. Let's see, I was checking the voltage at the source, where it was generated so let's go back to the EPROM programming pin itself. .012mv. (essentially zero). Ok, I'll go backwards toward the source to see if I can find out where the voltage isn't getting through.

Tracing the circuit diagram on the schematic showed that the voltage was controlled in a network of transistors. Now that's just great. I can just follow signals through logic chips, but my understanding of transistors is a bit sketchy. The principle is clear enough from a design point of view, but when it comes to what particular kinds of transistors do and how they connect into the circuit, that's something I don't really understand.

So, I says to myself, let's look at the logic portion of the circuit first. The way this particular PROM burner card works is as follows. The programming voltage is turned on and then the byte to be programmed is written to the EPROM. A timing circuit holds the write condition by telling the 8080 to WAIT using an extended wait-state. After each byte is written a number of times, so that the total amount of writing time for each byte adds up to the time the manufacturer of the EPROM requires for programming, a read condition turns off the programming voltage.

So I found a logic line entering the transistor network, and reasoned that that line turned on the write condition. So, I got out my logic probe and ran the program on an empty socket and looked at this logic line. It seemed to be working ok.

Now I don't have any choice. I've got to look at the transistors. I can't use the logic probe for that because the voltages are too high. Even though I don't know how each of these transistors work, there is only a couple of lines going in and out of the network. It turned out that all the transistors were of the same type. I checked my spare board and discovered that it generated the voltage ok, but didn't read memory correctly. So, I figured these transistors were ok. Well, I've got some good ones, but how do I figure out which one is the bad one (or if there is more than one)?

Ok, let's look at the EPROM voltage specification for that pin and see what it requires. A low voltage enables the EPROM for reading. And, the high voltage (26.5) is used to program it. The circuit looked like the high voltage came into one transistor and its output went to the programming pin of the EPROM. Ok, that makes sense, a hefty voltage being switched on and off by a transistor. But, there were two other transistors, both connected to the first one. It looks like the other two turn on the first one and also drain off any "leakage" from the main one. Just a guess. Well, we aren't getting anything through, so let's just try them one at a time.

The board was capable of programming 2716 eproms also, and had other transistors

of the same type for that purpose. I decided to simplify the board and remove the components used in programming the 2716's; this way, I could use those transistors on the 2708 circuit. Well, to make a long story short, this time I got lucky. The first one I replaced made the board work properly. Of course, Now that I think about it more carefully in writing this, I realize that the other components I had been removing might have had a short to ground which just kept the voltage down. I'll never know for sure. But one thing I did know. The high voltage chip got VERY hot before I got the board working again. I couldn't touch it. Now it just gets warm. So, your guess is as good as mine. It just goes to show that sometimes something can be fixed by someone who really doesn't know what he's doing.

We'll all need that kind of guts and luck to keep our Poly's running now that Poly has closed. Of course most of us stand ready to help anyone in trouble. And, most of the time a problem can be fixed by swapping boards or chips. But not always. Help keep our Poly's flying; share with us your down-time experiences so everybody can benefit from what each one of us learns.

True Circle

True Circle is a comedy booking agency in western Massachusetts that uses a Poly to support their business. As a newcomer to computing, the kinds of things being done are still simple. The proprietor, Norman LaFoe (Norm) reports that he has gotten past the initial "computer phobia" and is really starting to enjoy it.

To see how the Poly helps him, let's talk briefly about what the business involves. As a comedy booking agent, Norm arranges for comics to play a "gig" at various clubs in New England. To insure that a comic isn't used too often, detailed records of who played which club are required for immediate reference, particularly during the scheduling process. This is where the Poly makes things easy.

Norm has created a schedule file which is kept in date order and lists the date, the club, the comic, the number of shows performed on that date, and the salary. This file is updated each week by adding the data for the latest gigs with the system editor.

To use this data file during the scheduling process, Norm uses Abstract Systems Search.GO program, which will read in a memory full of data and display lines with the search string in it on the screen, paging when a full screen is achieved. Norm will give Search.GO the name of the club and it will display for him the past schedule of comics which played that club. He will give Search.GO the name of a comic and it will display for him the past gigs that comic has played, and will show the fee paid. The raw data for all past gigs is loaded into memory and can instantly display requested data. Norm says: "It's a real boon to the scheduling process."

True Circle keeps two mailing list files. One is for press release people, such as the local Television stations, Radio stations, newspapers, etc. The other is a list of regular patrons of the clubs who like to receive mailings telling of upcoming comics scheduled.

Another way the Poly has been useful is in the printing and preparing of directions to the clubs. These are sent to the comics to tell him or her how to find that particular club. "The directions alone have paid for the Poly. I save a bundle in long distance phone calls by mailing printed directions in advance.", reports Norm.

Edit Secrets

I have been working on Edit.GO and here are more "secrets" that I have found out. The ESCape library routines allow comments, direct text input, and immediate command execution.

An undocumented feature of Edit is that comments can be edited into ESCape libraries. This can be used as a title line to tell something about the file of commands, or as a comment about a particular escape command, or for any purpose whatsoever. When Edit is loading an ESCape library (ESC CTRL-L), it looks at the incoming text for the start of a definition. A definition starts with ^[= and the character being defined, and ends with ^[; The sequence '^["' marks the beginning of a comment. A comment ends with a another '"'. So, when Edit is loading commands and comes to '^["', it begins skipping text until another '"' comes along.

One minor annoying things about Edit is that it has no merge function. It only allows one to add from a new text input file; but, then one must go and mark the text and then move it to the desired location. Lately, I discovered a way around this. The ESC library load function will load the text and insert it where the cursor is. However, this routine has its own side effects; it strips carriage returns from the input text, and converts any character preceded by a carrot '^' into a control character. And, to get a single carrot, two must be in the file. If the control character is an Edit command it

will be executed just as if you had typed the instruction from the keyboard.

This allows for an interesting situation. One can load and execute the equivalent of a command file of Edit commands from within Edit by loading it with the ESCape library load function (ESC CTRL-L). I just tested a file into which I put the following:

```
In this demo file I am^M
TESTING the escape loader:^M
Let's see, we go to the beginning,^M
then we do a find TESTING,^M
then we do a CTRL-V to make the right^M
side of the line upper case,^M
then we do 7 backward arrows to move to^M
the beginning of TESTING, and finally^M
we do another CTRL-V to flip the^M
line to lover case.
^B^FTESTING^[^V^T^T^T^T^T^T^T^V
```

I have an application in which this text load feature comes in handy. I use graphics commands to my printer to create special logic, mathematics, and other graphics characters with my revised formatter. For example, I needed to use the Hebrew letter Aleph to represent cardinal numbers. When I first wrote the document I used an ordinary English letter, so I can easily read what I have written. But when it came time to print the document, I converted these into graphics instructions using the "graph" command that I have added to format. [See the article on GRAPHICS in this issue. Ed.] So, I create a sub-directory of files, each of which has one of the graphics character definition in it. Then I can go to the English character in the file I am editing, delete it, and do an ESC,CTRL-L and give the name and extension of the file for the graphics character I am using. This loads the graphics character command "on the spot" where the cursor is. I used to do this by defining an ESC library of the graphics characters, but Poly's Edit has a bug in it that writes over the beginning of the file in memory when there isn't enough room for the entire file. (SEE Bugnote 39.1 [HELP BUG EDIT-5]) Also, I didn't want to have to load graphics.ED, enter one character, and then reload TXdef.ED every time I wanted to replace one graphics character. Well, here's what's in aleph.TX:

```
{cmt "aleph" ,graph 0 0 2 68 71 117 116 100
8 8 8 9 85 87 114 50 32,cnt}
```

When my formatter executes these three commands it produces the following output.

א

The "cmt" command allows inserting comments in a format file. The "graph" command interfaces to the IDS Prism and Dataproducts Paper Tiger printers dot graphics mode. The same effect can be achieved with the "chr" command. [See GRAPHICS elsewhere in this issue for more on graphics mode printing. Ed.]

Well, there are other ways this feature can be used. For example, Suppose you use the Poly to write contracts which have a lot of standard clauses to pick and chose from. A "boiler plate" document could be written with a marker, say three asterisks ('***'), where various clauses would be inserted. A separate file could be written for each of the clauses which would be inserted at the various markers.

With this setup, the operator would edit the boiler plate document giving a new output file name for the finished contract. Then, using the CTRL-F routine, find a marker, delete it, and use ESC CTRL-L to load the desired clause at the appropriate location.

Another way this feature could be used is by programmers when writing programs. Commonly used subroutines could be put in individual files. Then when one of these subroutines is desired during the process of writing a program, the ESC, CTRL-L procedure could load the desired subroutine at the desired location.

The immediate command execution feature can be used give Edit the equivalent of an INITIAL file. Here is my first idea of how to use this. Suppose we always put the date on the first line in our text files, and that we have a date program on the disk which keeps the current date in a file called "Date.DT". If we EDit TXdef.ED and move to the end of the file, we ought to be able to insert a direct execute routine which deletes the old first line and inserts the date. First we would need to move the cursor down one line, delete the line, and then get the date from the date file. ^R^X^M^Q^[^LDate.DT^M. Well, it turns out that this doesn't work. The Edit ESCape library routines do not allow loading a library from within a command. So, that leaves this idea out. But we could use the routine to find a particular string of text, such as the word VERSION. That would be a gentle reminder to change the version number each time we edited a program. At the end of the BSdef.ED file we could insert ^FVERSION^[^S^S^S^S^[^H. This would cause Edit to find VERSION, move right 4 characters and then start the cursor blinking. Nice try, but this doesn't work either. Edit loads the ESCape library BEFORE it loads the text file, so there is nothing to find at the time of loading. But, can we load plain text? Yes, that does work. You can put messages to yourself which get inserted into the

text whenever the library loads. However, once EDit has been started and you are already editing text, the ESCape library load routine will execute commands

Graphics

Many modern printers are capable of "dot addressable graphics". My IDS Briter Writer, IDS Prism, and Dataproducts Paper Tiger printers all allow dot addressable graphics. In dot addressable graphics, the printer is switched to graphics mode with a command sequence, which may vary from printer to printer. Once in dot addressable graphics mode, each byte of data sent to the printer encodes a column of 7 dots (vertically) to be printed. Most printers transfer data in 7 bit serial with an optional parity bit; but, some can transfer data in 8 bit serial or in parallel too. The printers above can all be configured for 7 or 8 bit parallel mode as well as for serial mode.

Each "byte" of data consists of 8 "bits"; for example, the character '1', has an ascii (decimal) value of 49, but its hexadecimal value is 31H. A hexadecimal number is in base 16, which is very convenient for writing the value of a byte, because 16 is an exact power of 2, and each byte takes just 8 bits with no left-over, unused numbers. Writing each bit as a '0' or '1' express it in "binary" notation. The character '1', which has an ascii value of 49 decimal, or 31H (hexadecimal), has a binary value of 00110001B. Each hexadecimal digit converts directly to 4 binary digits.

A serial data transmission line sends these bits out one at a time over one wire; a parallel data transmission line sends these bits out at the same time over 8 wires, one for each position.

Well, it doesn't really matter how they get to the printer, but once they are there, the printer can use them to tell which dots are to be printed and which ones are to be left blank. In my printers, the lowest order bit "addresses" the top dot in a column of 7 rows. Each higher bit turns on or off the correspondingly lower dot. For example, a binary 0000-0001 (I'll put in a '-' to make reading these numbers easier.) turns on the top dot. A binary 0000-0010 turns on the second dot, and a binary 0000-0011 turns them both on. This same logic applies to the remaining 5 bits. Here are what some (decimal) number bit patterns would look like when printed in graphics mode. The number at the top of each column is the decimal value of the byte of data being sent to the printer, and the asterisks show which bits are turned on or printed. I have numbered the bits '0' to '6' instead of '1' to '7' because of the relation between the bit number and the powers of two. Here's what the first 9 patterns, and a couple of more, look like

```
bit - 1 2 3 4 5 6 7 8 9  16 31 32 44 63 64 100 127
  0 - * . * . * . * . *   .  *  .  .  *  .   .   *
  1 - . * * . . * * . .   .  *  .  *  *  .   .   *
  2 - . . . * * * * . .   .  *  .  .  *  .   .   *
  3 - . . . . . . . * *   .  *  .  *  *  .   *   *
  4 - . . . . . . . . .   *  *  .  .  *  .   .   *
  5 - . . . . . . . . .   .  .  *  *  *  .   *   *
  6 - . . . . . . . . .   .  .  .  .  .  *   *   *
```

Important ones to notice are the ones with only one asterisk. These tell us what number turns on which bit. Here they are again. Also, I show a diagram with the bit number replaced by the bit value. (The value of this diagram will become apparent later.)

```
bit - 1  2  4  8 16 32 64        value - 1  2  4  8 16 32 64
  0 - *  .  .  .  .  .  .            1 - *  .  .  .  .  .  .
  1 - .  *  .  .  .  .  .            2 - .  *  .  .  .  .  .
  2 - .  .  *  .  .  .  .            4 - .  .  *  .  .  .  .
  3 - .  .  .  *  .  .  .            8 - .  .  .  *  .  .  .
  4 - .  .  .  .  *  .  .           16 - .  .  .  .  *  .  .
  5 - .  .  .  .  .  *  .           32 - .  .  .  .  .  *  .
  6 - .  .  .  .  .  .  *           64 - .  .  .  .  .  .  *
```

To get any of the other bit combinations printed, we just need to figure out which bits we want to turn on, and then add up the corresponding bit values. In the previous diagram, notice that the column under the number '3' has both the 0 bit and the 1 bit turned on. The value of these bits is 1 and 2, respectively, and that ads up to 3. Similarly, the bits under the 31 have bits 0, 1, 2, 3, and 4 turned on, and the corresponding bit values, 1, 2, 4, 8, and 16 add up to 31. To make character design easy, I created a template with these bit values -- but first, a bit more on my printers.

My Paper tiger is capable of a very dense graphics character display with 168 columns of graphics dots per inch. This is twice the dot density of the Prism printer, which is limited to 84 dots per inch in graphics mode. And, the Briter Writer is so old that I didn't write my program to support it. Actually, all three printers were made by the same company, who changed their name in mid stream. The later printers are "upward compatible" with earlier printers.

Most of the time I use graphics, it is to print a special character not available on my printer. To make the process easier, I created a graphics character design grid. My original graphics character designs were done on the Briter Writer, but ran perfectly on the Prism, and later on the Paper Tiger. (Actually, IDS had a Paper Tiger printer before the Prism, but when IDS was absorbed by Dataproducts, the name

"Paper Tiger" was used again. -- The IDS Paper Tiger is earlier than the Dataproducts Paper Tiger.)

So, when I was updating my graphics characters to take advantage of the Dataproducts Paper Tiger's 168 dpi capability, I set about preserving the same character's "look" on the Prism. Since the Paper Tiger could print twice as many dots in the same space as the Prism, I decided to design my character so that every other dot column could be printed on the Prism and get the original character, but that extra dot column could be inserted between each column and "fill in" the character a bit. After some experimenting, I decided on to use 17 column of dots for a Paper Tiger character and 8 columns for the Prism. (Of course "8" shuffles into "17" with a perfect 1-1 interleaf.) I drew a blank character matrix with the dot number and value on the left, and a 't' or a 'p' at the top of each column so I could see what the character would look like before it was printed. Here is that blank character matrix.

```
     !tptptptptptptptpt!
0  1 !.................!
1  2 !.................!
2  4 !.................!
3  8 !.................!
4 16 !.................!
5 32 !.................!
6 64 !.................!
     !tptptptptptptptpt!
```

To give you an example of how I use this matrix, here is one filled in for the Greek character "alpha".

```
     !tptptptptptptptpt!
0  1 !.................!
1  2 !.................!
2  4 !.................!
3  8 !......****..**...!
4 16 !....**....**.....!
5 32 !....**....**.....!
6 64 !......****..**...!
     !tptptptptptptptpt!
```

The Poly's Video Terminal Interface (VTI) uses a 9x7 character display format on an 10x15 field. (The first dot column is blanked to form the intercharacter separation. The bottom 6 rows are blanked to form the lines between characters.) So, adapting the 9x7 character format to a printed 7x8 or 7x17 format requires a little compromise and adjustment. Some of the Poly's Greek characters can be created using the same dot pattern. The alpha above is one of them. Of course, it will be a little taller (in proportion to other characters) than it would appear on the Poly screen. I always used the first one or two columns as inter character spacing, and I kept the same number of columns, to fit the character into "fixed" character spacing.

Anyway, back to designing the characters. The next step is to add up the value for each column using a 0 where there is no asterisk and the bit value where there is one. For the "alpha" character this comes out to be the following numbers. The second line lists every other number to show what numbers would be used by a printer with only 84 dpi capability.

```
0 0 0 0 48 48 72 72 72 72 48 48 48 48 72 72 0.
  0 0   48   72   72   48   48   72
```

Some time ago I changed FORMAT.GO by added the "chr" command and calling the result "format.GO". Since then format.GO has been extensively modified by adding many other commands, including one which shifts into and out of graphics mode directly. That new command is "graph" (what else?). Two other new commands are relevant to the current discussion, the "cmt" command and the "cnt" command. The "cmt" command allows inserting comments in text to be formatted, comments which will be skipped over. In BASIC we have REM statements which get skipped over; In Exec (and in assembly language programs) the semicolon is a remark or comment command. Well, I wanted a comment command for the formatter and added one to format.

The other command, "cnt", "counts" invisible or graphics characters; it basically tells format to shorten its idea of how long the line is by one character. When graphics characters are created, format doesn't know how to count them so doesn't figure them into the character count. That could result in the right margin being shifted over one character on a line with a graphics character in it. I invented the "cnt" command to deal with this situation. It adds one to the line character count so format will start a new line after the right number of characters have been added to the line to be printed. So, when I took the above numbers and put them together with the graphics command I got " {cmt "alfa",graph 0 0 0 0 48 48 72 72 72 72 48 48 48 48 72 72 0,cnt}" The cmt command ignores the space and "alfa". The comma marks the start of the next command "graph", which switches the printer into the graphics mode, and then, depending whether the "prism" or "tiger" has been selected, sends the numbers following direct to the printer; when the numbers are exhausted, "graph" switches back to text mode. Finally, the "cnt" command "counts" the graphics character "∝" as part of the line.

Now, if you have an early version of format or Format, which only has the "chr" command, the same effect can be achieved,

but one must know what the graphics and graphics escape command sequence are. In the case of the Prism and Paper Tiger printers, (and many other printers), that sequence is just an ascii 3 to enter graphics mode and an ascii 3 and 2 to exit graphics mode. Because ascii 3 signals a command, to print a graphics 3 requires ascii 3 followed by another ascii 3. All this has been written into the graph command. But here is how we would do the same thing with the "chr" command. I will start with the lower density graphics which uses only 8 columns per character. The first "chr3" shifts into graphics mode, and the final "chr3,chr2" exits. 84 dpi graphics "alpha" using the "chr" command: {chr3, chr0, chr0, chr48, chr72, chr72, chr48, chr48, chr72, chr3, chr2}. The 168 dpi graphics version would be: {chr3, chr0, chr0, chr0, chr0, chr48, chr48, chr72, chr72, chr72, chr72, chr48, chr48, chr48, chr48, chr72, chr72, chr0, chr3, chr2}.

Of course, one only has to do this once. Then we store the result in an escape library as a set of definitions that can be called up and inserted with two key strokes.

If your printer is only capable of 84 dpi graphics, then you might want your character matrix and alpha to look thus:

blank 84 dpi matrix

```
    ¦ p p p p p p p p ¦
0 1 ¦ . . . . . . . . ¦
1 2 ¦ . . . . . . . . ¦
2 4 ¦ . . . . . . . . ¦
3 8 ¦ . . . . . . . . ¦
4 16¦ . . . . . . . . ¦
5 32¦ . . . . . . . . ¦
6 64¦ . . . . . . . . ¦
    ¦ p p p p p p p p ¦
```

alpha 84 dpi matrix

```
    ¦ p p p p p p p p ¦
0 1 ¦ . . . . . . . . ¦
1 2 ¦ . . . . . . . . ¦
2 4 ¦ . . . . . . . . ¦
3 8 ¦ . . . * * . * . ¦
4 16¦ . . * . . * . . ¦
5 32¦ . . * . . * . . ¦
6 64¦ . . . * * . * . ¦
    ¦ p p p p p p p p ¦
```

If you only have Poly's FORMAT.GO program it is possible to do the same thing using the backslash "\" character. But, the character count on the line could get screwed up and one would have to do certain shenanigans to get the appropriate characters needed. To do the alpha character requires the following tricks. I'll just do the 84 dpi version to keep it simple. First, we need to note that chr3 is a CTRL-C, and chr2 is a CTRL-B and chr0 is a CTRL-@. But, we can't just type these characters and get them because Edit will think they are commands. CTRL-B goes to the beginning; CTRL-C continues the last search, and CTRL-@ is ignored by the ROMS! Here's what we have to do to get a chr0. First we define the key 0 with the sequence ESC,=,0,^,@,ESC. This will cause the Greek letter alpha to appear whenever we type ESC,0. Ok, now for the other CTRL characters. We can get them in the same way, or we can use CTRL-F in combination with CTRL-U. First type CTRL-F. You will be rewarded with a double cursor. Then type CTRL-C,CTRL-C,CTRL-B and ESC. First you will see two Greek letter delta's and the Greek letter gamma, and then they will all disappear. Without moving the cursor, type 4 CTRL-U's. You will get the brick and these three characters back. Three left arrows later the cursor will be to the right of the brick. How can we tell? They both look the same. If you type one DELETE, the two bricks will become 1 if the cursor is to the right of the undeleted brick. By the way, that brick is chr127 which you might need at some other time. Anyway, now that you have these three characters, position the cursor between the two deltas. The first delta will put the printer into graphics mode and the second one together with the gamma will take it out of graphics. That takes care of the chr3 in the beginning and the chr3,chr2 at the end of the line. Now we must look at the others. There are two chr0's so using our ESC,0 twice we get them. Next, the chr48 is the same as the character '0' so we type that. The chr72 is a capital 'H' so we type that twice. Two more 0's and one more H and we are almost done. Now we move the cursor in front of each character and insert a backslash "\". The backslash will tell FORMAT.GO to print the next character literally. I don't know if this will work for the alpha character. You may need to go into the file with Szap and change the 00 bytes to 80H bytes. It doesn't work with format, and, to my recollection, I never changed the routine that gets characters from the file, so 00 Bytes get skipped over. One caution. Edit removes any 00 bytes from a file when it is loaded, so if you EDIT the file again, you will need to go through the process of putting in the alpha character again (and perhaps use Szap to change the 00 bytes to 80H too.)

More on Modems, etc.

by Bob Bybee

Dear Ralph,

Thanks for the article about modems and data communications. As you know, it's always been my primary field of interest. Since you brought up the subject of early modem software on the Poly, I'll respond with some (slightly autobiographical) information.

My first Poly was an "orange toaster," the Poly-88, which I bought used in 1979 for $300. It came with keyboard and monitor, video card, and CPU card with 512 bytes of RAM. The box had no other memory. I was interested in learning computers, though, so that didn't slow me down for a minute! I used the "4.0 Monitor" ROM listings that came in the Poly's manual, and learned 8080 machine language. I began writing some small programs in machine code

too, but I did it the hard way: by hand-assembling the programs. I would write out my program on paper, then write down the machine-code bytes in hexadecimal that corresponded to each instruction, and finally I'd type in those bytes through the Front Panel.

When I hit "G" to run the program, either I had entered everything right, or it would usually erase itself and I'd have to start over! Of course, since I had no disk or tape on which to store the program, I'd have to leave it in memory with the power turned on, or else key it in again next time.

Well, I didn't write very long programs in those days. A few hundred instructions was about my limit. But I did manage to write a small communications program, which let my '88 talk to a 300-baud modem. That was my first entry into Poly comm software. I used it to communicate with The Source, a time-sharing service that I and several other Poly owners (Frank Stearns and Chuck Thompson, among others) belonged to.

Eventually my Poly grew up. I got a printer (a 30 character-per-second GE TermiNet sort-of letter-quality printer) and added a "real" chassis, and my '88 cards moved into their shiny new home, a North Star "Horizon" box. I added memory and two floppy drives, and I was in heaven! Then I started more serious software work.

A local writer, Stuart Woods, used to do all of his work on a Poly. Through Mark Sutherland and PolyLetter, I met up with Stuart, and even interviewed him for a PolyLetter issue once. He had acquired a communications program from PolyMorphic which was a derivative of FTP, Poly's File-Transfer Program. This program would send and receive text files, and didn't care if it was talking to FTP at the other end of the wire, so it was useful for Stuart; he could send text to his publisher's computer. I got a copy of the program from Stuart. He had told me that Poly specifically disavowed any knowledge of it and would not support it, so he asked me to do some work on it.

And work I did. Ralph, using your DisAssembler, I tore into the program and learned what made it go. Then I started adding features, like the one that captured incoming data into a file. I made two versions of the program, as you noted in your article... MODEM-T for text, and MODEM-M for machine code. I used these programs for several years, and sold them to a number of Poly owners using the company name Smyrna Software. Yes, it was an ugly name, but I lived in Smyrna, Georgia at the time, and even National Geographic couldn't make Smyrna pretty.

After becoming tired of MODEM-T's limitations, I began designing a new communications program called SM. By this time I was comfortable with all aspects of programming the Poly, and ready to build a "really neat" program. SM included features for autodialing on the Hayes Smartmodem 300 and similar modems, as well as many other advanced things like: XMODEM file transfer protocol, automatic login scripts, and a user-extensible command language. As modems got faster, I improved SM to work at higher baud rates. When directly connected to another computer (using wires, not modems) SM can transfer files at 9600 baud, over 3 sectors per second.

SM was probably the most popular program I ever sold on the Poly. In recent years, I've continued to ship a few, mainly to people who want to transport files from the Poly to an IBM or other system. SM can transfer files to/from a PC, Macintosh, or any other system equipped with a comm program that speaks XMODEM (most do). SM is still available, by the way!

I shipped the source code with early versions of SM, as a service to users who had been burnt by software developers that went out of business. While I had no intention on leaving anyone without support, some customers felt more comfortable if they had the source, and could take it to a programmer if it someday needed modification. I stopped shipping source with the later versions of SM, since the source no longer fit onto a single disk! (It's a big program.) Still, I made it available to anyone interested.

A few last notes to clarify items in your article. Most modems can send and receive at the same time; they are called "full duplex." Both ends CAN talk at once, and in fact, I've often done "type-ahead" when dialing into a time-sharing system. I can type to it while it's typing to me.

Your description of how a modem works is correct, up to the 1200 baud variety. Beyond that, more exotic modulation schemes are used, and a baud is no longer equal to a bit per second! How is that possible? Well, a "baud" (after Mr. Baudot, whose name was also given to a teletype coding scheme) is loosely defined as a "change in state, per second." But by careful encoding, one "change in state" can be used to transmit more than one bit of information. At speeds of 2400 baud, a single shift in the phase of the signal actually conveys TWO bits of information, called a "dibit". At high speeds, modems are more accurately rated in bits per second, or bps, not baud.

It takes fancier hardware to detect and

generate these subtle phase shifts, and that's why modem technology stalled at the 1200 baud speed for several years. Today, however, high-speed modems use digital signal processing (DSP) technology to detect and generate the signals that they put on a phone line. DSP has been the reason for the drop in 2400-bps modem prices. It allows all of the complex filtering to be done with digital circuits, not analog ones. Analog filters are touchy and must be built with expensive, high-precision components.

At 9600 bps, an even fancier modulation scheme is used. It's called "trellis coding," and I can't pretend to understand it completely. Let's just say that it uses the techniques of changing a signal's phase AND amplitude (size), in various ways, to transmit more than two bits with each change of state.

In addition, any of these modem techniques can be speeded up even more using data compression. Compression reduces the redundancy, or unneeded information, that's present in most data. This reduces the number of bits required to hold that data. And since transmission time is directly proportional to the number of bits we send, that time is reduced too. Modems which use compression must also "decompress" at the receiving end, so the bits at the other end look like they did when they started.

There are a number of compatibility problems between high-speed modems. First, not all of them use the same compression technique (some use no compression at all). When two modems can't agree on a compression technique, they can still send data, but in uncompressed form that takes more time to send. There is another form of incompatibility that's even more severe, and that happens when two modems (usually the 9600 bps kind) use different modulation schemes. This would happen if the two modems failed to agree on the frequencies used to transmit over the phone line. In that case, the two modems can't talk at all!

This kind of standardization problem used to exist in the 1200-baud world too... remember the old Bell 212 vs. Bell 202 conflict? Both were 1200-baud modem types, but they were incompatible. Eventually the 212 standard won out, since it was full-duplex and 202 only supported half duplex (transmitting one way at a time). A similar war is in progress in the 9600-bps arena now. But, I doubt that many PolyLetter readers will be buying 9600-bps modems anytime soon. They're still very expensive.

You can achieve some of the same speed improvements by doing your own data compression, before transmitting a file, and uncompressing it at the other end. In the PC world there are several programs which perform data compression and "archiving," or combining several related files into one large file. Most of the PC bulletin-boards refuse to carry anything BUT archived files. These can be downloaded more quickly than the original files, and then you can uncompress them at the receiving end. Often a set of files can be compressed by more than 50% using these procedures, which saves you a lot of money in long-distance telephone bills, and saves the BBS operator a lot of disk space. Some IBM-PC programs that do compression are SEAware's ARC, and Phil Katz' PKARC and PKZIP programs. I know of no such program on the Poly, though.

[Bob Bybee can be reached at Poly Peripherals, 5011 Brougham Court Stone Mountain, GA 30087, (404) 498-3556. -- Ed.]

SORT.GO

I have found and corrected a bug in my SORT.GO. The nature of the bug is that an extra character is added to the first line in memory whenever there are blank lines in the input file. The bug can be avoided by just making sure that there are no blank lines in the input file. However, Anyone who wants the upgrade, send in your disk and $1 for postage and handling and I'll send the corrected replacement along.

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. Noncommercial ads by subscribers are free.

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.

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

DISKS - MODEMS - PROMS - SOFTWARE - SPELL

1. MAXELL diskettes: 5-1/4" 10 hard sector -- $15 per box.
2. Used diskettes: 5-1/4" 10 hard sector -- $0.50 each.
3. Hayes Micromodem 100 (300 baud S-100 internal modem) $30.
4. HayesSys modem software (for the Micromodem 100) $30.
5. Abstract Systems Exec (Enhancements & bugs corrected) $35.

6. Abstract Systems Proms (Enhancements & bugs corrected) $35.
7. PolyGlot Library Volumes: $6 each; 5 or more - $5 each.
8. Hayes Smartmodem 1200B (IBM compatible internal) $60.

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

In the Public Domain

PGL-V-31 and PGL-V-32 contain Abstract Systems Spelling Checker. These two disks can be used as is to spell check a text file. If you have drives with a larger capacity, you can copy all the Spell files into a subdirectory named "SP.DX" and use SpellS.GO. Spell.GO looks for dictionaries on drive '?' SpellS looks for dictionaries in subdirectory '<?<SP'.

Included on the disk is HELP.GO and HELP files for Spell support programs. Alpha.GO is a program to alphabetize a dictionary file. Reverse.GO is a program which sorts a dictionary into the order as if the word had been spelled in reverse. This feature is useful when looking for words with the same suffix. $12 for the pair of disks.

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 PROGRAM COMPARE
        HELP file for system Program "COMPARE".

COMPARE.GO compares two files, byte by byte, that are specified
on the command line. If there are any mismatches, it displays
the byte count from the start of the file and the two bytes
that differ. It asks whether the comparison is to be sent to
the printer in addition to the screen.

Syntax: "COMPARE <n<path1<file-one[.EX] <m<path2<file-two[.EX]"
Example "COMPARE SORT.GO <2<SORT.GO"
```

In This Issue

Coming Soon

Relocatable Code, Poly Meta.

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

Address Correction Requested

FIRST CLASS MAIL

Ralph E. Kenyon, Jr. EXP: 9999
Abstract Systems, etc. 184
191 White Oaks Road
Williamstown, MA 01267-2256

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 ads are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* (1980 thru 1988) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 - $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 ($3.50, $4.00, $5.00).

PolyLetter

Editorial

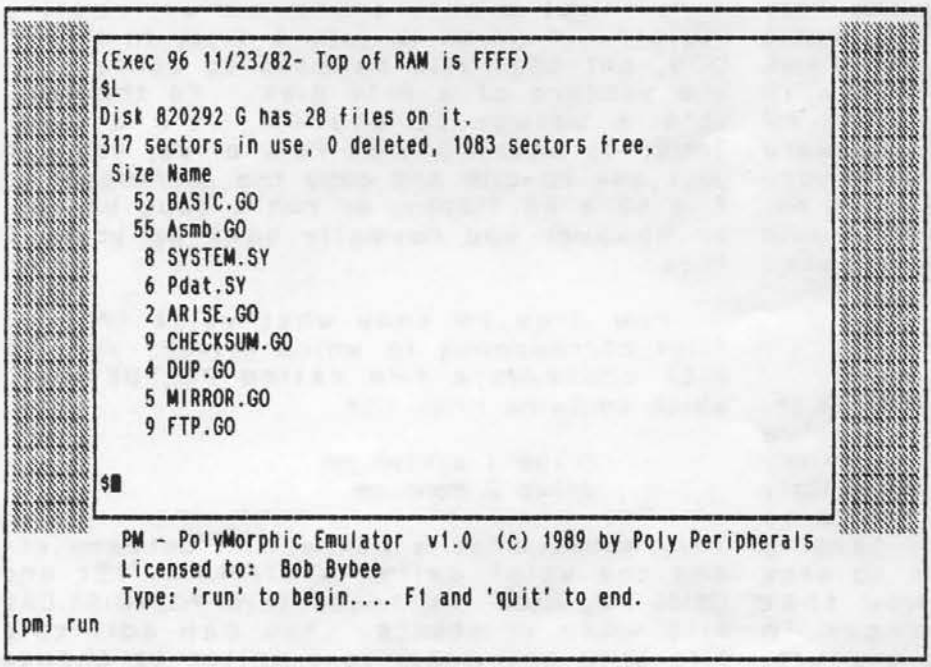
This issue is devoted principly to a special event in the Poly world. Bob Bybee has just released a Poly Emulator which runs under MS-DOS. What this thing can do is just amazing; what it can't do is understandable. First I'll let Bob brag about it, then I'll tell you my reactions. There are a few trade-offs, but all in all, it seems to be a great deal. There's going to be life in Poly software for a while yet! Old Poly's never die, they just compute away, and Exec lives on forever! Imagine -- having Edit and FORMAT and Poly BASIC in the PC world!

Poly Emulator

Poly Peripherals introduces PM, a PolyMorphic Emulator program for the IBM-PC and compatibles!

With PM, your PolyMorphic software investment is still viable. PM runs nearly all Poly programs, BASIC or assembly language. It will boot any version of Exec. PM provides a complete Poly environment, and even runs the Front Panel and RDB debuggers. PM provides 56K of RAM space, so your Top of RAM is FFFF.

PM uses "virtual disks." Each Poly drive (1 through 7) is emulated by a PC-DOS file. Disk access is much faster than Poly disks ever were, plus you can use DOS facilities to make backup copies. Utilities are available to transfer files between the Poly and IBM worlds.



[This is what the IBM-PC screen looks like with the Poly Emulator running on it. The border marks off 64 characters wide by 16 lines. The bottom portion of the display shows the messages from the emulator. The Poly system runs in the 64 by 16 portion of the display. Since IBM-PC graphics are different, programs which use Poly graphics don't display correctly. Also, since 8080 machine code is emulated on the IBM-PC, the emulator may run a bit slower than the Poly. If you have a fast, or turbo XT or AT, the difference in speed won't be seen as much. For more information read EVAL.EM -- Ed.]

PM runs on a PC, XT, AT, 386, or 100% IBM-compatible machine. For more information, or for a free demonstration disk, contact: Bob Bybee, Poly Peripherals, 5011 Brougham Court, Stone Mountain, GA 30087, (404) 498-3556.

Announcements

Left Coast Software has just announced that it is tossing Exchequer, Version 2.1, into the shareware arena with a $15.00 registration fee. [Exchequer was reviewed in the JAN/FEB 1989 issue of PolyLetter. Ed.] Left Coast has arranged discounts on check purchases from Moore, NEBS, and Delux. The shareware registration fee should be sent to Left Coast Software, P.O.Box 160601, Cupertino, CA 95016-0601. Left Coast accepts telephone orders using VISA, MC, and COD at 1-(408)-996-3130. A trial evaluation copy may be obtained from PolyLetter for $5, or call me up and I can send the files from my PC using a 1200 baud modem (your nickel).

Poly Emulator

by Bob Bybee

PM - PolyMorphic Emulator - User's Manual
Version 1.1, October, 1989 Copyright (c) 1989 by Poly Peripherals

Introduction.

Welcome to PM, the PolyMorphic Emulator program for IBM-PC compatibles. PM is part of our continuing effort to keep your Poly systems working for your benefit. PM allows you to run your existing Poly software on any IBM-PC compatible system, which gives you the benefit of readily available service, parts, and backups.

PM does just about everything the Poly does. PM provides an complete PolyMorphic environment, in which any Poly programs will run, whether they were written in BASIC, assembly, or other languages. PM accomplishes this by emulating the hardware of the Poly, including the ROMs, memory map, interrupts, screen memory, and so on. Anything that runs on Poly hardware should run on PM, with a few exceptions noted below.

Why Emulate?

If you're a loyal, long-time Poly user, you've reached 1989 painfully. You've spent a great deal of time and money developing programs to run on your Poly system, and you've bucked the trend of IBM-PCs for years. But with every passing year, it's becoming more difficult to stay with the Poly, especially so now that PolyMorphic Systems is no longer in business. You're worried about support, and about getting spare parts and service. Someday the last PolyMorphic will stop running.

But software doesn't die of old age the way hardware does. With PM, you can move virtually all of your Poly software to a PC-compatible, and continue using it. And, if you want to, you can use the utilities available with PM to begin modifying your programs and data to run under MS-DOS.

What You Get

On the IBM-PC disk, you'll find these files: PM.EXE is the PolyMorphic Emulator program. POLYDEV.DAT is a text file that sets up your default drive configurations (discussed below under vdisks). SYSTEM.PM is the System vdisk, which is a bootable Poly System disk. It contains Exec/96, BASIC, and many utility programs. DEMO.PM is the second vdisk, containing several demo programs.

If you purchased the transporter kit (optional), you'll also receive an SSSD Poly diskette of SM, Poly Peripherals' communications program. This version of SM has been enhanced so that it can send an entire Poly disk all at once, to a PC which will capture it into a file. You'll also find a copy of PROCOMM on your IBM-PC diskette. PROMCOMM is a communications program which is used on the PC side to capture this data from SM. And, if you ordered it, you'll find a serial port cable custom-made to connect your PC to your Poly.

The Vdisks

When running PM, you can access drives 1 through 7, just like normal. But each of the "drives" you are accessing is actually a "virtual disk", which we will call a "vdisk". A vdisk is just a file, in IBM-PC DOS, but that file happens to contain all the sectors of a Poly disk. To the Poly, it's a drive; to the PC, it's a file. Thus, to back up that Poly drive, you can just use PC-DOS and copy the corresponding file to a PC floppy, or run a tape backup, or however you normally back up your PC files.

How does PM know what vdisk (PC-DOS file) corresponds to which drive? Your PC disk contains a file called POLYDEV.DAT, which contains lines like

```
drive 1 system.pm
drive 2 demo.pm
```

This establishes a connection between <1< and the vdisk called SYSTEM.PM, <2< and DEMO.PM, etc. PM reads this POLYDISK.DAT file when it starts. You can edit this file with any ASCII text editor to change your defaults. You can also use the PMU program, described later, to change these connections "on the fly", just as you could with Poly's Volume Manager.

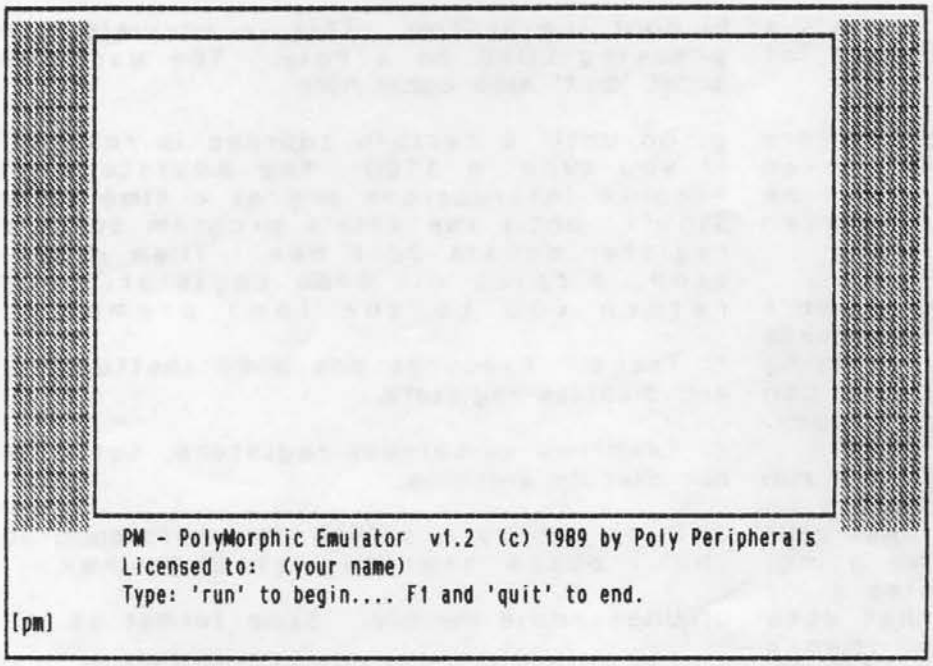
Speaking of Volume Manager: Don't use it

with PM. Its functions aren't needed with PM, and it contains hardware-dependent code that won't run correctly. Also, it occupies RAM space, whereas PM can do all of the same tricks without moving Top of RAM below FFFF. Use POLYDISK.DAT and the PMU program to do your volume-manager functions.

Getting Started

If you have a hard disk, create a directory for PM to run in, such as C:\PM. Copy all files from the PM IBM-PC disk into this directory. If you don't have a hard disk, you should consider getting one! But you can run PM from a floppy drive. Please make a backup copy of the floppy disk before doing so.

Type PM and hit Enter. You'll see this screen:



The upper window is the Poly screen, 16 x 64 characters. The lower window is used when you enter "supervisory" commands to PM, to start and stop the emulator, for example.

Type RUN (or R) and hit Enter. The Poly Emulator will boot up. If you see the message "Waiting for printer..." on the lower window, hit CTRL-Y to bypass the printer initialization, just as you would on a Poly. If you see that message, your printer was probably not connected or turned on when you told PM to run.

After several seconds, you should see the Exec prompt in the upper window. (If you're using a PC or XT, be patient. It'll take a little longer than you're used to.)

Now, anything you type should work just like a Poly. You can list drives, edit a file, and so on. Since your POLYDEV.DAT file was set up with two vdisks, you can do things with drives <1 and <2 now. Later we'll explain how to use other drives.

Keyboard Mapping

You may want to leave your CAPS LOCK feature turned on when using PM, since most Poly commands are upper case. Unlike the Poly keyboard, you can use SHIFT while CAPS LOCK is on, and get lower case.

PM lets you use your PC keyboard in ways that are useful in the Poly environment. Here are some of the control-keys, and what they do: DEL and the BACKSPACE key (the little left-arrow by itself) both do a Poly DELETE, which is used almost always as a backspace. The four arrow keys work as expected. The PgUp and PgDn keys perform ^P and ^N, which the Poly editor uses for previous and next page. Ctrl-PgUp and Ctrl-PgDn are ^B and ^E, which the editor uses for beginning and end of file. F1 through F4 are Poly I through IV (only found on the Poly Keyboard III). These are equivalent to ^\, ^], ^^ (CTRL-^), and ^_ (CTRL-DELETE). F5 generates the same code as the Poly "Line Feed" key, ^J. F6 generates the same code as the Poly "Backspace" key, ^H. All of the other keys should generate standard characters, or not do anything at all inside of PM. Other function keys and special keys may be defined for future enhancements.

Printer Support

The PM system vdisk is mostly a "plain vanilla" Poly disk. PM is able to boot from any Poly disk; we didn't modify Exec in order to get it to run. But this vdisk has a special Sio.PS file, which is the PM printer driver. This file must be on any vdisk you plan to use as a System disk. If you want to use one of your own disks as a System disk, you'll need to remove Sio.PS from that disk, and copy ours onto it.

PM only supports printers when using a version of Exec that contains Sio.PS. Earlier Execs had the printer driver embedded in Printer.GO, and these Execs cannot be used with PM if you want to use a printer.

When running Setup, you can give any legal answer to the Baud Rate question, such as 9600. It doesn't matter to our Sio.PS file.

When you tell a Poly program to print something, it will come out on whatever

your PRN or LPT1 printer device is. If you can do a print-screen on your PC and get a printout, you should be able to get a printout from PM.

If you're using a serial-interface printer with your PC, be sure your system sets up its baud rate with the MODE command, and redirects LPT1 to COM1 so that the print-screen test works. If you need more information, see your DOS manuals for the MODE command, or call us for help.

Note: The version of Sio.PS shipped with PM version 1.0 (the demo version shipped free to some Poly owners) would "hang" if the printer was not ready. This has been fixed in PM/1.1. Do not use the System vdisk from PM/1.0 unless you copy the Sio.PS file from the 1.1 release.

The Buts

There are always exceptions. Here's a list of some known things PM can't do (at least in this release).

Block graphics and some Greek characters don't come out right, since the PC screen can't display anything resembling those characters. See comments under Screen Mapping, below.

The Poly serial port hardware isn't emulated, so programs like FTP that access the serial port directly won't run. Using our Sio.PS printer driver, though, you can use any printer that your PC normally uses.

Some speed-dependent programs won't run correctly. The speed of PM is dependent on what type of PC you're using, and what your programs spend most of their time doing. In general, a program that computes a lot will run slower, but a program that does lots of disk I/O will run faster, than a real Poly system. Use the fastest PC you can get. Running PM on a 286 or 386-based PC is about as fast as a real Poly. On a PC or XT, it's slower, but it does work.

Screen Mapping

The Poly's screen is 16 x 64 characters. PM displays this in a window of the PC's 25 x 80 screen. The PC can't display all of the Poly characters accurately... some of the Greek characters are missing, and none of the "block graphics" characters will look exactly right. If there's enough demand for it, this problem could be cured by some additional work on PM, and by adding a "programmable font" display card to your system, such as the Hercules RAMfont card. Contact Poly Peripherals if you're interested in having a 100% accurate Poly character set.

Commands at the [pm] Prompt

When you start PM, the lower window shows a prompt of [pm]. At this prompt, you can type 'run' or 'r' to start running PM.

To get back to this prompt, hit ALT-F1. Then, to quit PM, type 'quit' or 'q'. For best results, don't do this unless you're at the Exec prompt, with the $ or $$ showing on the Poly screen. If you interrupt the program while it's doing something like updating a disk directory, you could damage some Poly files.

If you've interrupted the emulator with ALT-F1, you can continue where you left off by typing 'r' again.

Some other commands you can type at the [pm] prompt:

b: Boot the system. This is equivalent to pressing LOAD on a Poly. The warnings about 'quit' also apply here.

g: Go until a certain address is reached. If you type 'g 3200', the emulator will execute instructions one at a time (very slow!) until the Poly's program counter register equals 3200 hex. Then it will stop, display all 8080 registers, and return you to the [pm] prompt.

t: Trace. Executes one 8080 instruction, and displays registers.

x: Examines (displays) registers, but does not execute anything.

d: Dump memory. 'd 2000 20' would dump 20 (hex) bytes starting at 2000 hex.

u: Unassemble memory. Same format as 'd'.

The PMU Program

PMU is the "PM Utility" program. It handles things which involve the PM environment, or interactions between the IBM-PC and the emulator program. It's also the PM substitute for Volume Manager.

To run PMU, just type its name at the Exec prompt:

```
$PMU
pmu:
```

When you see the "pmu:" prompt, type in any of the PMU commands described below. If you don't have this document handy, type HELP to get a brief list. Commands may be entered in upper or lower case.

Show

The SHOW command lists the current PM

vdisk assignments. Example:

```
pmu: SHOW
Current drive assignments:
1:  system.pm
2:  demo.pm
3:
4:
5:
6:
7:
```

Connect

The CONNECT command changes a drive assignment. Type the CONNECT command, followed by the Poly drive number, then the PC filename for the vdisk. For example:

```
pmu: CONNECT 3 STUFF.PM
```

would assign Poly drive <3 to the PC file STUFF.PM, which should be a Poly vdisk. If it isn't, you'll get the message "disk directory destroyed!" when you try to access it.

Size

This command changes the size of a Poly vdisk. (Try that, Exec!) You can increase the size until it uses all available disk space (or 65535 sectors at most). You can also shrink a vdisk until it only has room for the files currently stored on it (and therefore has 0 sectors free). Example:

```
pmu: SIZE 3 500
```

If you request a disk size that's smaller than the number of sectors in use, PMU will complain. If you request a size that's larger than the amount of free space available on your PC disk, PMU will increase the size as much as it can, using all that available space, then will tell you what the size was actually set to.

When you create a vdisk, by sending data from a Poly to the PC over a serial port, that vdisk is created with 0 sectors free. You will probably use the SIZE command to increase the vdisk's size before doing much with it.

When using SIZE, keep in mind how you plan to back up your Poly vdisk files. If your PC uses 360K diskettes, you can create a vdisk of about 1400 sectors and still copy that file to a diskette. With 1.2 meg diskettes, you can copy a vdisk of about 4600 sectors. Anything larger and you won't be able to make a backup with a simple PC "copy" command. You'll need to use a backup procedure on your PC that splits files among diskettes, such as the PC's "backup" and "restore" commands, or else use a tape drive as your backup device.

DOS

The DOS command suspends operation of PM and takes you to a "DOS shell". Use this to perform a few simple DOS commands, look around at files, etc. But don't plan to stay here for long; PM is still loaded in memory and may interfere with any large programs you run while in the DOS shell. If you plan to do much with DOS, exit PM with ALT-F1 and Q.

When you're done with the DOS shell, type EXIT to return to PM.

Copy

The COPY command moves files between the Poly and IBM-PC file systems. You must specify the source and destination filenames. For example,

```
pmu: COPY <1<POLYFILE.TX
C:PCFILE.TXT
```

would copy the Poly file <1<POLYFILE.TX to IBM-PC file C:PCFILE.TXT. You can copy in either direction, from Poly to PC or vice-versa.

You must include a path specifier on either the Poly filename (like <1<), on the PC filename (like C:), or both. Otherwise, PMU can't tell which is which, and will complain. For best results, don't put \ or : in your Poly filenames, since it will make them look like PC filenames.

PMU will then ask,

```
Text or Binary treatment (T or B) :
```

Press T or B, and then Return. If you select text treatment, the file is copied with the proper end-of-line characters (CR alone on the Poly; CR/LF on the PC) and the NUL bytes at the end of the Poly file are stripped out. If you select binary treatment, all bytes are transferred.

Remember that Poly filenames are case-sensitive (upper and lower case are considered different), but PC filenames are not.

Quit

The QUIT command exits PMU and returns you to Exec.

Moving Your Disk Files

PM doesn't really become useful until you can run YOUR programs in the PM environment. In order to do this, you'll need to transport your disk files from your "real" Poly, to the PC. Poly Peripherals provides ways to do this.

The simplest way is to have us do it for you! If you can put your Poly files onto 5" SSSD Poly diskettes, and send them to us, we will transport those files onto IBM-PC diskettes and send them back. Each Poly disk you send to us will be converted into a single IBM-PC disk file. We'll name them with names like DISK1.PM, DISK2.PM, and so on. Then, when you receive those files on PC disks, just copy them into your C:\PM directory (or wherever you plan to use PM). You can "connect" any of those vdisks to any PM drive number using the PMU program. Once connected, you can copy files around until you get the Poly disk arrangements you want.

If you want to be able to transfer Poly files yourself, you need our "transporter kit." This contains everything you need in order to transfer whole Poly disks into IBM-PC files. You'll be doing it just the way we would, but you can do it yourself whenever you want to.

To use the "transporter kit," copy all PROCOMM files from the IBM-PC disk provided, into your C:\PM directory:

```
C:\PM>copy a:procomm.*
```

Then run PROCOMM on the PC, by typing PROCOMM and pressing Enter. PROCOMM comes with a large documentation file, which describes it in great detail. It's a powerful package, useful for general purpose communications work on any PC. PROCOMM is "shareware," not public-domain, so if you intend to continue using it you are asked to "register" your copy with the authors by sending them a fee. You can also purchase a commercially available version of PROCOMM, with improved features, at many software stores.

This set of PROCOMM files is configured for transferring files from the Poly. It is set up to use 9600 baud, 8 bits, no parity, one stop bit. Later, we'll describe how to tell PROCOMM to receive files.

But first we need to tell the Poly how to send them! On the Poly disk provided, you'll find a copy of SM/3.4, the latest version of Poly Peripherals' communications program. This, too, is pre-configured for the purpose of transferring disks to the IBM. To use SM, copy both the files SM.GO and sm.DT to your System disk.

Verifying the Connection

Connect your PC's serial port to the Poly's port with the cable provided. Go to the PC, and execute PROCOMM by typing its name. On the Poly, run SM by typing its name. Both systems should be set to 9600/8/N/1. Use Alt-P on the PC, or CTRL-DELETE on the Poly, to see the current settings.

If everything is set properly, you should be able to type a few characters on the Poly keyboard, and see those characters on the PC screen. If you type on the PC, it should come out on the Poly. If either of these tests fails, your systems aren't set up properly, or there is a cable problem. You can't transfer files until you pass these tests.

Transferring

This version of SM knows how to send an entire Poly disk using XMODEM protocol. To use this feature, press CTRL-DELETE to bring up the SM menu. Insert the Poly disk you want to send, into one of your drives. (If it's a hard disk volume, you should have already connected it using Volume Manager before executing SM.)

On the PC, press PgDn (for download). Select XMODEM protocol, and enter a filename (we suggest something like DISK1.PM - using .PM as the extension on all your vdisk files makes it easier to tell what they are). The PC will then wait for a file download in XMODEM format.

Go over to the Poly, and (at the SM menu), type the command XSDISK n, where n is the drive number you want to send, 1 through 7. (XSDISK stands for "XMODEM send disk"). To send the disk in drive 2,

```
XSDISK 2
```

The two systems should immediately begin churning as the Poly sends, and the PC receives. When the process is complete, you are the proud owner of a file on your PC, and this file can be used as a Poly vdisk by PM. All the Poly files on that vdisk are now safely stored on your PC.

History Of 8080 Emulation

When Intel developed the 8086, 8088, and other similar processors, they gave some thought to compatibility with the 8080 CPU. Not much thought, but some thought. It turns out that for each 8080 register, there's an 8086 register that does almost the same things. And for most 8080 instructions, there are comparable 8086 instructions. But the opcode values are entirely different. For example, a NOP (no-operation) is 00 in the 8080, and 90 on the 8086.

A few years ago, NEC (Nippon Electric Company, a large Japanese electronics firm) developed the V20 processor chip. This was a plug-in replacement for the 8088 used in the IBM-PC, but with the addition of an "8080 emulation mode." They were able to

switch their processor into, and out of, a mode where it would actually execute 8080 opcodes. Unfortunately for the V20, IBM-PCs became obsolete and were replaced by ATs and 386 systems, and there is no V20 that will plug into those machines. Scratch one 8080 emulation mode.

Still, several software packages became available which used the 8086 to emulate an 8080 (or a Z80) in software. Since the 8086 can't directly execute 8080 opcodes, each of these programs had to use an "emulation loop" that works something like this:

- get the next opcode,
- execute it,
- and repeat.

Where the 8080 would execute one instruction during this procedure, the 8080-emulator program would have to execute many 8086 instructions to (1) get the opcode, (2) advance the PC, (3) figure out what to do to emulate this instruction, and (4) do it. This means that an emulator generally runs much slower than the original processor. PM uses a number of optimization techniques, and runs at about the same speed as a "real" Poly when used on an 80286 or 80386-based PC. We consider this to be a success.

These 8080-emulator programs typically used the CP/M operating system. CP/M has several features which make it adaptable to the emulator approach:

- CP/M is fairly independent of the hardware it runs on.
- It has a simple memory map (all RAM from 0000 to FFFF).
- It can run without interrupts.
- Many CP/M systems use diskettes that an IBM-PC can read.

None of these things applies to the PolyMorphic! The Poly has a more complex memory map, with ROM, disk controllers, memory-mapped screen, and heavy use of interrupts. A good Poly emulator would have to duplicate all of these things. PM is the result of several years of research into this problem, and does duplicate all important features of the PolyMorphic architecture. It won't allow you to read Poly disks in PC drives (yet!), but we have other ways to move your data into the PC.

Some of the inspiration for PM came from the the "Portable 8080 System", by M. Sekiguchi, of the Japan C Users' Group. This and other wonderful C code and information is available from the C User's Group, 2120 West 25th Street, Suite 8, Lawrence KS 66046, (913) 841-1631. The 8080 system was on their disk number 284, and is an 8080 emulator developed entirely in C.

Technical Details

This section describes how PM was created, and some of its internal workings. You do not have to read or understand it in order to use PM effectively.

In this document, we'll refer to all of the IBM-PC type processor chips as the 8086, even though some PCs use an 8088, 80286, or 80386 chip instead. For our purposes they are all equivalent, except in terms of speed.

Most of PM is written in the C language, using Borland's Turbo C compiler. The high-speed portions are written in 8086 assembly language using Turbo Assembler.

The register mapping between the 8086 and the 8080 emulator is similar to the V20's emulation mode mapping:

8080	8086
F, A	AH, AL
H, L	BH, BL
B, C	CH, CL
D, E	DH, DL
SP	BP
PC	SI

Notice that the 8080 AF register pair is stored backwards, in AH/AL. The 8086 provides the SAHF and LAHF to move the 8086 flags register to/from the AH register. So to emulate an instruction that will update one or more flags, we must move the AH register into the 8086 flag register; do the instruction (which will update the 8086 flags); then move the 8086 flags back into AH. For example, the code for INR B looks like

```
Op04:   sahf            ;Opcode 04: INR B
        inc     ch
        lahf
        jmp     Endop
```

Why not let the 8086 flags retain the state of the 8080 flags at all times? The emulator does some other work during each loop, which modifies the 8086 flags. To keep the 8080 flags accurate, they have to be squirrelled away safely in AH.

Flags

Intel was kind enough to provide a good mapping between the 8080 flags and the 8086 flags. The 8086 has a 16-bit flag register, but the low 8 bits of this duplicate the 8080 flags exactly. But what about the instructions that modify those flags? In most cases, there are 8086 instructions which do exactly what the corresponding 8080 instructions do. A notable exception is the DAD instruction, which updates the C flag only. The emulator contains special code to handle these situations.

Interrupts

PM emulates the keyboard, real-time clock, and single-step interrupts on the Poly CPU. The real-time clock was a particular problem, since the IBM-PC has a clock that interrupts at 18.2 Hz, but the Poly's clock interrupt is 60 Hz.

The solution was to reprogram the PC's clock to interrupt at 182 Hz. This is then divided down by interrupt service routine code, by 10 to drive the PC's 18.2 Hz needs, and by 3 to provide about 60 Hz for PM.

I/O Ports

PM emulates only a few essential Poly I/O ports. IN 18 gets the keyboard code after a keyboard interrupt. OUT 0C enables the single-step interrupt. Emulation of other I/O ports may be added in the future.

System Calls

There are many functions that the emulator can't perform directly, such as disk I/O. For this reason, PM defines a "system call" opcode which was unused in the 8080 opcode list. If the emulator sees an opcode in the form ED ED nn, it skips the two EDs, and uses nn as a "system call" value. At that point, PM stops emulating opcodes and goes into a system call handler which can perform the desired function, based on the value of nn and the 8080 registers.

When writing programs to run under PM, do not use any system call opcodes. You could crash the system or cause damage to your files. We do not document the available system calls.

Screen Handling

There are two ways to write to the Poly screen. Programs can either call the screen driver routine in the ROMs (through WH1), or write directly to screen memory. PM supports both.

When going through WH1, PM uses a system call (described above) to go back to the 8086 and let it do the screen writing. This is much faster than letting the emulator execute the dozens of instructions that the ROMs use for screen output, even though that method works too (and was used in a pre-release version of PM).

PM also copies the Poly screen memory to the PC screen, many times each second. Anything written to screen memory by any method will show up on the PC screen almost instantly. This method does sometimes cause a strange "jumping" appearance on the screen, if PM chooses to copy the screen before the Poly program has finished updating something. (Anyone who has used DESQview on the PC will see similar effects, since it does the same trick.)

EVAL.PM

I was in communication with Bob about the emulator after looking at the demo version. I made several suggestions to him, which he has incorporated into the commercial version. Bob has incorporated the Abstract Systems enhancements to the Poly Proms into his Emulator. Those enhancements are:

1. RIwe (Read line with editing) now supports CTRL-U similar to Edit.GO. This enhancement allows RIwe to return the character under the buffer pointer whenever CTRL-U is pressed. The main advantage is that the command line can be restored (and edited) for immediate re-execution. Any program using rIwe will then have an immediate re-execute capability.

2. Change DSPLY to WH1 in the following routines:

```
CROUT   JMP WH1 vice JMP DSPLY
CLEAR   JMP WH1 vice JMP DSPLY
BLK     JMP WH1 vice JMP DSPLY
TABBER  JMP WH1 vice JMP DSPLY
HEXO    JMP WH1 vice JMP DSPLY
```

These enhancements prevent "holes" in "Printer LOG" output when these routines are called by programs.

One feature the demo version did not support was Serial i/o. Bob has written that into his commercial version, but the underlying PC facilities are dismal; the emulator implementation is only as good as those underlying facilities. The first time I tried to send characters to the emulator from the Poly it locked up! Closely tracing the problem showed that the CTS line on the RS232 port was never raised. So, the Poly's 8251 USART chip couldn't send anything. (Of course this is cable and header plug implementation dependent). If your plugs are wired like Poly said, there might not be a problem. I wire mine all crossed on the header so the cable can be straight through. Well, the long and the short of it is that no signals got through because the 8251 didn't see a CTS (clear to send) signal. It turns out that the problem is not the emulator's fault. The fault lies deep in the pc bios implementation. The bios routines never turned CTS on to receive a character!. It only turned on DTR.

Since then I have written a small TSR program for the PC which takes control of the serial i/o interrupt and turns on CTS as well as DTR. Unfortunately the pc

implementation is very primitive. It operates the 8250 asynchronous receiver transmitter chip in a polling mode. This is like having a mailbox with room for ony one letter; when the postman comes to deliver another, he throws away any letter that was still in the mailbox! Actually, Poly's implementation of the Serial port driver was similar, except it allow room for twenty or so characters. At any rate, because the emulator runs slow and there is a lot of software overhead, the risk of losing characters is very high. The Poly can easily overrun the emulator.

Bob's transfer method is to send from the Poly to Procomm running on the pc. I assume he runs things at 1200 baud, since that is the speed defined on his printer driver for the IBM-PC and his program SM.GO is designed for use with 1200 baud external modems. Once Procomm received the file, Bob uses PMU.GO to bring the pc-world file into the Poly Emulator world [see the manual]. This transfer process is apparently not perfect, as one of the five disks I sent to Bob had holes in the files. I suggest you run CHECKSUM.GO on any disk to be transferred, and then run CHECKSUM.GO inside the emulator on the resulting file, just to make sure the transfer was correct. I have not yet talked to Bob about what might have caused the problem. I'm sure he'll eventually find out the cause and correct it. (Bob thinks the problem is with some difference between 48 tpi formats written by 96 tpi drives, but this is the first time anyone has reported problems since I converted many years ago.)

Of course, Bob reports in his manual that the system runs slower, and that it doesn't do graphics right. -- How much slower? I used the following benchmark program to time it.

```
10 INPUT "How big shall I try? ",N
20 DIM A(N)
30 PRINT "Press any key to start.",
40 Z=INP(1) PRINT
50 MAT A=RND(100)
60 PRINT "Done."
```

By running the same program on both the Poly and the emulator I was able to compare the actual speeds. It turns out that on an IBM PC or compatible running at 5.77 MHz, the emulator runs at about 1/16 the speed of the Poly. When I set my turbo XT up to 8 Mhz, the ratio was reduced to about 1/13. Now, these figures apply only to so-called "compute-bound" jobs, and reflect the difference in speed for programs which spend a lot of time just computing. When it comes to disk I/O, the problem is a little different. The pc disks step faster and the data transfer rate is higher. I am unable to make a real comparison because my system uses drives which step faster, but by running a disk emulator which is designed to run a SA-400 drive at the normal step time, the Poly took 66 seconds to Sniff 260 sectors while the emulator took only 24 seconds on the floppy. (Poly 96 tpi drives took 47 seconds to sniff the same amount.) This is a net speed up of 2.75, assuming my rigged up setup runs at the same speed as a "normal" Poly. The Poly would be much faster with double density drives. Someone with regular Poly drives could make a comparison one way by Sniffing a disk. On my system with a hard disk, the comparison is Poly - 12, Emulator - 28, so the Emulator takes 2 1/3 times as long to sniff the same amount of hard disk space as the Poly does. The upshot of this is that you can expect to get an emulator performance index somewhere between .0625 and 2.75, depending upon the proportion of computation and disk i/o. One thing I did notice, is that I can type faster than Edit running on emulator can take the characters; any good typist will experience the same annoyance (at 8 Mhz or less).

The Other Guys

Poly's COMP-DISK program compares two disks. On the PC and compatibles, the program DISKCOMP.COM does the same.

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 BUG EDIT-5

Abstract Systems BugNote 039.1 April 19, 1985

Edit 4.0 (11/12/82)

Edit 4.0 has a bug in the escape library loading routine. Loading a large escape library after editing has begun and memory is nearly full results in writing on top of the text in memory! Edit does not report that there is not enough room for the escape definitions; it writes the definitions over the top of some of the text in memory.

To avoid this bug load escape libraries before filling up memory with text. If you know you are going to need to load a large escape library, load that library before starting to edit 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. Noncommercial ads by subscribers are free.

Entire 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.

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

DISKS - MODEMS - PROMS - SOFTWARE - SPELL

1. MAXELL diskettes: 5-1/4" 10 hard sector -- $15 per box.
2. Used diskettes: 5-1/4" 10 hard sector -- $0.50 each.
3. Hayes Micromodem 100 (300 baud S-100 internal modem) $30.
4. HayesSys modem software (for the Micromodem 100) $30.
5. Abstract Systems Exec (Enhancements & bugs corrected) $35.
6. Abstract Systems Proms (Enhancements & bugs corrected) $35.
7. PolyGlot Library Volumes: $6 each; 5 or more - $5 each.
8. Hayes Smartmodem 1200B (IBM compatible internal) $60.

(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 litter bit to ponder. Cats like two litter boxes; one for each output.

In This Issue

Coming Soon

Relocatable Code, Poly Meta. More: BASIC for Beginners, System Programmers Notes, Help, BugNotes, Public Domain Software, etc.

Questions

Can you find and answer the questions asked in this issue? Send your answers and requests in. Did you see the movie Short Circuit? Need !INNPUT! Send me something to print!

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

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 ads are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* (1980 thru 1988) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 - $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 ($3.50, $4.00, $5.00).

PolyLetter

Editorial

This issue opens a new era in PolyMorphic Systems computing. With the release of Bob Bybee's Poly Emulator program for the PC and clones, our favorite software, System-88, is no longer dependent wholly upon PolyMorphic Systems hardware. PolyLetter welcomes System-88 users running the emulator on IBM compatible machines.

I will put the PolyGlot library disks on IBM format disks as they are requested in that format. Abstract Systems Exec will now be available on IBM format disks for emulator users.

Someone out there didn't get his or her last issue of PolyLetter. Good 'ol Uncle Sam's Postal Service sent one back with the address label ripped off. They had the nerve to ask for a better address! Well, I don't know whose it was, but I do know it wasn't one due for renewal. Let me know who you are and I'll have the replacement in the mail pronto.

Drive Service

Poly drives can be repaired and aligned by ISR. They report they can handle as few as one or as many as hundreds of all types of floppy drives. Infinite Service & Repair (ISR), 8104 Parkdale Drive, Austin, TX 78758, 1-800-458-6778.

Letters

PolyLetter January 5, 1989
I would like to see more information for the beginners. Thank you. -- Arnell Baker, Guadalupe, CA.

In the Public Domain

I am still going over many old disks received from Poly, but haven't got anything new to report at this time. However, if you want any of the PolyGlot library disks on IBM-PC format for use with Bob Bybee's Poly Emulator system, let me know and I will have them converted.

One Day in the Life of (AIS) Poly

Once upon a time there was a Poly at PolyLetter. Every day it did some things the same and some things different. "What did it do the same", you ask? The first thing she does is to connect the volumes according to the default values set by V-SETUP in Vmgr.OV. Once these are set, a new drive becomes the boot drive and Poly opens INITIAL.TX. The first command is to SQUEAL. That makes sure I know whenever there are soft disk errors on the floppy drives. The second command is to ENABLE. That insures that my listing shows system and deleted files. After that, Poly goes to subdirectory "u" (for utilities) and runs the date program. The date program immediately turns off command file mode, looks for the file Date.DT on the system drive and displays that date on the screen. It then asks if the date is correct. The date program looks for an ESC, an "N", an "n", or anything else. If I were to type ESC, which I sometimes want to do, date.GO will abort command file mode and return control to Exec. If I type either an "N" or an "n", date.GO asks me to input the correct date.

It does this using RIwe and by pointing the buffer pointer at the existing date. Because of the "get back" feature built into the ASROM proms, I can type CTRL-U and get back one character at a time up until I get to the day which may be wrong. Then I can type the correct day of the month, and continue with more CTRL-U's until the rest of the date has been "gotten back". A final carriage return (or the one gotten by the last CTRL-U) sends date.GO scurrying to write the correct date back to the file Date.DT. Of course, date.GO writes zeros after the date so as to erase any remains from longer dates that might have been

left. Once date has written the file back to disk, it restores command file mode and turns control back to Exec. She then reads the next command in INITIAL which is RL<Clock.

Clock.RL is actually a relocatable file and cannot be executed directly, but this Poly's Exec is smart; it recognizes the extension ".RL" and goes out and gets the relocatable loader program AKA LoadRel.GO. I taught Exec to do this in the same manner that Exec knows how to get BASIC when in finds a file with the extension ".BS". LoadRel expects a .RL program to be A high-memory TSR program. Like BASIC, the relocatable loader loads the program in high memory and then executes it. Once Clock gets control, it turns command file mode off and asks what time it is. When I answer, Clock begins ticking off the time in the upper right corner of the screen, in the format HH:MM:SS. Clock then turns command file mode back on and returns control to Exec, which is getting commands from INITIAL.

The next command is RL<Gnomus.RL, which is the memory resident keyboard macro program. Like Clock, Gnomus is loaded by the relocatable loader and executed. All Gnomus does is set up its buffers and return control to Exec. She gets the next command in INITIAL, which is RL<edit.RL. edit.RL is the screen editor, another TSR program; edit justs hooks itself into wh0 and gives control back to Exec.

The next command in INITIAL is the back-space character followed by the CTRL-L character and the letters "WP<Gn" The backspace character is Gnomus's hot key character. This activates Gnomus. The CTRL-L tells Gnomus to load a file of macros. "WP<Gn" tells Gnomus to load file "Gn.Gn" from subdirectory "WP" on the system drive. This loads into Gnomus's internal buffer the macro commands I regularly use. Once Gnomus loads the file, it returns control to Exec, which is still getting command from INITIAL.

The next command in INITIAL is "u<Wait", which tells Exec to execute the program Wait from subdirectory "u". Wait turns off command file mode and waits for an actual key-press. If that key is ESC, then Wait aborts command file mode and returns control to Exec with the rest of INITIAL aborted. This is what happens normally when the computer is booted for the second or more time on the same day. If any other key is pressed, Wait resumes command file mode, giving control back to Exec and the next command in INITIAL.

The first cold boot of the day requires some other programs to be run. The next command in INITIAL is BASIC BS<DATE.BS. This saves a little bit of time by saving Exec from having to call Gfid twice before giving control to BASIC. When DATE is run, it reads the contents of Date.DT, converts its form to another format and updates WP<Date.DT. DATE.BS then returns contro to Exec (and INITIAL).

The next command in INITIAL is RL<PramKey. PramKey is a temporary TSR program which stays in memory while command file mode is running, but clears itself out when the command file is finished. The purpose of PramKey is to retrieve a parameter from the keyboard buffer for another program while command file mode is running. Once PramKey is executed it hooks into WH0 and watches for the character "%" That character is replaced by a string taken from the keyboard buffer. The next act of INITIAL is to Type the date with the system command "T Date". This causes the date to be displayed on the system screen (where edit can get at it). The next character in INITIAL is the CTRL-DELETE character (function key IV) which is the hot-key character to activate edit.GO CTRL-DELETE is followed by a throw-away space. This causes edit to move up to the tail end of the line with the date that was just printed on the screen. At this point, edit is activated and the incoming characters are interpreted as edit commands. The next character is an ESC character which tells edit to execute an escape sequence command. Following that is the up-arrow character which tells edit to move to the beginning of the line being edited. The next character is the carrot "^" which is edit's control character prefix. (It converts the next character into a control character.) The very next character is an "H", which edit converts into a CTRL-H (the back-space) character and inserts it on the line in front of the date. The next two characters, "=d" are entered on the line following the backspace character and in front of the date. Following this is an ESC to cause edit to execute another escape sequence. The escape character is followed by a

down-arrow character which instructs edit to move the cursor to the end of the line. Next is another carrot, edit's control character prefix, followed by the left square bracket "[", which edit converts into an actual escape character to append to the end of the line. So, now we have a backspace, equals, d, the date, and an escape on the line, which just happens to be exactly the sequence to activate Gnomus, telling it you want to define "d" as the value of the date. The next character in INITIAL is a carriage return, which instructs edit that the current line being edited is finished and should be copied to the keyboard buffer for execution. But, since we are in command file mode, the next thing to be executed is coming from the command file. That just happens to be the character "%". Remember PramKey? It's still active and watching WHO, so when this character is brought in by the command file, it replaces it with the stuff in the keyboard buffer up to the carriage return; which just happens to be the backspace, equals, d, the date, and an escape. The backspace activates Gnomus; the equals says to define a key; the d says which key, and the date is what Gnomus defines that key to be. The escape finishes the definition. Of course, PramKey restored input from the command file, so the next command comes from INITIAL. That is the command "; Today is " followed by the backspace character and the letter d. This causes the comment "; today is ", to appear on the screen. The backspace activates gnomus and the d tells gnomus to spit out the macro replacement, namely, the date it was just defined as. So, I get "; Today is " and the date neatly displayed on the screen. The final command in INITIAL is "BASIC BS<TICKLER", which activates the tickler and followup program. TICKLER.BS turns off command file mode and reads the datafile TICKLER.TX for items which are past due, or coming due in the next week. (I maintain this file with Edit.) TICKLER reminds me that PolyLetter is past due, and that the mortgage payment is coming up, and that various other things need attention.

Primitive I/O in IBM BIOS

Well, I have finally begun to dig into the IBM and have been working on my first assembly language program for the clone. While evaluating the Poly Emulator [PL 89/5] I tried to send text to the PC at 9600 baud without losing characters; it didn't work. At first I blamed the Poly Emulator, but when I looked closer, I found the fault was with the underlying pc-bios. I once had a similar problem with the Poly.

Long ago, I reported a bug in Poly's Sio.PS. When I first began transferring text files back and forth between the Polys, I noticed that the Poly would lose characters. Normally FTP was used to transfer files between Poly's and the problem doesn't show up under this condition. I was experimenting with something more sophisticated, like one Poly giving orders to another. The receiving Poly was losing characters from its input stream. Since the Poly's were connected via the serial ports, I began looking at the Sio driver. A close look at the source code for Sio showed the problem. BugNote 17, on April 11, 1983 reported that Sio just threw away characters when its input buffer filled up.

I wrote a custom version of the serial device driver for the Poly, which I call sio.PS, and which can tell the sending computer to wait when its input buffer fills up. It does the job by dropping CTS when there is only space for 5 more characters in the input buffer. I use an improved version of that driver when inputting characters from my pc-clone; it prevents the clone from overrunning the Poly's capability.

Recently, I have begun to experience the reverse effect. The Poly overruns the clone, even at 8 Mhz! When I transfer files to Procomm on the clone, I usually don't have any problems. Apparently Procomm runs fast enough to gobble characters at 9600 baud without any problems. I did discover that when Procomm was downloading directly to a floppy, the slower disk speed caused some loss. I bypassed this by only downloading to the hard-disk. Recently, I tried to transfer a LONG paper and discovered some losses even when downloading to the hard-disk. To overcome the problem, I wrote a BASIC program to run on the Poly for sending text files to the clone.

```
10 DIM A$(1:255),F$(1:31) \FILE:2,LIST
20 INPUT "File to transfer to the clone? ",F$
30 FILE:4,OPEN,F$,INPUT
40 READ:4,A$ \IF A$<>"" THEN 49 ELSE N=N+1
46 IF N<10 THEN 40 ELSE 60
49 FOR I=1 TO N \PRINT:2,"" \NEXT \N=0
```

```
50 PRINT:2,A$ \GOTO 40
60 PRINT "Back to ",CALL(1027)
```

By running a BASIC program for file transfer, the Poly is slowed down just enough to prevent over-running the clone at any speed.

Of course, because the Poly Emulator runs so much slower, it was easy for the Poly to over-run the Emulator at any speed. Needless to say, this practically eliminates direct transfer from the Poly to the Poly Emulator, a prospect I found most annoying.

So, I began to look long and hard at the Clone's serial transfer routines. With the help of the IBM Technical Reference manual for the XT (which includes the bios source code listing) I discovered that the clone's serial character capabilities are really primitive compared to the Poly's. The clone does single character processing in what is essentially a polling mode. Even the earliest Poly proms had such a single character routine, but accomplished the transfer using interrupts. The polling was done by the wormhole 6 routine using a one byte flag and a one byte buffer for receiving a character from the usart. The usart interrupt routine was primitive indeed; it just stuffed the last character into the buffer, regardless of whether the previous one had been removed or not.

Guess what the IBM and compatibles do? You guessed it. They just leave the character in the input chip and the last one received is the only one available! The pc and compatibles do not even have interrupt level routines for serial data transfer! You might find such routines in programs which take direct control of the 8250 asynchronous receiver transmitter chip, but not in the bios routines. Incidentally, the 8250 is not capable of synchronous data transfer. It is strictly an asynchronous serial i/o device. The 8250 chip does have its own 16 bit on board baud rate divide logic, but it's really a less sophisticated device than the 8251. The 8251 depends upon external chips for its baud-rate, but is capable of synchronous data transmission and reception.

So, the primitive pc-bios routines make no allowance to tell sending devices to stop sending characters. It has no buffer for incoming characters. It depends upon the holding register in the 8250 chip itself to store the last character received until the clone cpu asks for it. Meanwhile, if another character comes in the old one just gets lost. Moreover there is no way for an external sending device to find out whether the clone has taken the character or not. No signal is used for this purpose. Using the built-in pc-bios routines, there is no way to reliably send characters to a pc-clone.

Once again, a pc-clone is a step backward from the Poly!

Relocatable Files

Readers have asked for an explanation of relocatable files. Here's the short of it.

Relocatable files are generated by the Poly assembler when the popcode RELOC is used. RELOC must appear before any code is generated and tells the Asmb.GO to generate a relocation word-map. The word-map is written to the file after any code and storage space is generated. Asmb then makes the Load address of the file point at the byte in the file where the relocation map starts. Asmb leaves it up to the operator to give the output file a ".RL" extension.

And, here's the long of it. Let's see what's involved in relocating a machine language program. Suppose you had a tiny little program which did nothing more than clear the screen. Suppose we call that program CLS and see what it would look like in assembly language.

```
        REFS SYSTEM.SY
        REF USER
        REF WH1
        ORG USER
        IDNT $,$
Start   MVI A,0CH       ;0CH is a form feed
        CALL WH1        ;clears the screen
        RET             ;Back to Exec
        END
```

Now this program is such that it can be loaded and run anywhere in the 8080 address space. That is because it does not CALL or JMP to any internal routines. It does CALL the system vector WH1, but that is fixed at 0C24H. Most programs do CALL or JMP to internal routines.

Suppose we look at another simple program, one which displays a sign-on message.

```
        REFS SYSTEM.SY
        REF USER
        REF Msg
        ORG USER
        IDNT $,$
Start   LXI H,Signon    ;Point HL at our message
        CALL Msg        ;Display message pointed to by HL
        RET             ;Back to Exec
Signon  DB 0CH,'Good morning.',0DH,0
        END
```

Now this program makes reference to an internal label, Signon, the value of which depends upon where the program is to be loaded in the 8080 address space. It will be 7 bytes past Start. Since USER is 3200H, Signon will be 3207H.

If the program is to be loaded and run someplace other than USER, then it must be re-assembled. In a very early version of Poly's System-88 the value of USER was 3400H. If this program was assembled to run under that system, then Start would have a value of 3400H, and Signon would have a value of 3407H.

Sometimes it is desirable to load a program up under MEMTOP so it can remain in memory while a USER program is being run. Poly's version of Gnomus, and Poly's early version of the debugger were such programs. One had to edit the source file, insert the new start address, and then re-assemble it. This process would give many different versions of the same program that all load and run in different memory sized systems.

What would be nicer, would be to have some way to load a program and run it at different memory locations, especially when we don't know how much memory the user will have. A natural solution is to keep track of all the addresses used internally to the program and to change these addresses to the correct value whenever the program is loaded at some particular memory address. The RELOC popcode tells the assembler to keep track of these addresses.

When RELOC is used in an assembly language program, Asmb makes a relocation map of all locations which much be changed when the program is relocated. Suppose we change the above program to be relocatable. The assembly language file would then look like this:

```
        REFS SYSTEM.SY
        REF Msg
        RELOC
Start   LXI H,Signon    ;Point HL at our message
        CALL Msg        ;Display message pointed to by HL
        RET             ;Back to Exec
Signon  DB 0CH,'Good morning.',0DH,0
        END
```

Notice, that we have replace the "ORG ..." and "IDNT ..." lines with "RELOC". A relocatable file isn't designed to live in any particular place in memory, so there's no ORG instruction. Here's what the output of the assembler looks like in the first case.

```
                        REFS SYSTEM.SY
                        REF USER
                        REF Msg
3200                    ORG USER
3200                    IDNT $,$
3200 210732     Start   LXI H,Signon    ;Point HL at our mess
3203 CD0C04             CALL Msg        ;Display message poin
3206 C9                 RET             ;Back to Exec
3207 0C476F6F   Signon  DB 0CH,'Good morning.',0DH,0
320B 64206D6F
320F 726E696E
3213 672E0D00
                        END
```

The directory entry will look like this:

```
 Size Addr   La   Sa flags Name.
    1 19F6 3200 3200     N RL.GO
```

The column under "La" (load address) and "Sa" (Start address) contain the address in the 8080 memory space where the program is to be loaded and run. Here is a dump of the actual contents of the file.

```
3200 ¦ 21 07 32 CD 0C 04 C9 0C ¦!.......¦
3208 ¦ 47 6F 6F 64 20 6D 6F 72 ¦Good mor¦
3210 ¦ 6E 69 6E 67 2E 0D 00 00 ¦ning....¦
3218 ¦ 00 00 00 00 00 00 00 00 ¦........¦
```

A relocatable file has no such absolute address. Here what the output of the assembler looks like in the relocatable case.

```
                        REFS SYSTEM.SY
                        REF Msg
                        RELOC
0000 210700     Start   LXI H,Signon    ;Point HL at our mess
0003 CD0C04             CALL Msg        ;Display message poin
0006 C9                 RET             ;Back to Exec
```

```
0007  0C476F6F  Signon  DB 0CH,'Good morning.',0DH,0
000B  64206D6F
000F  726E696E
0013  672E0D00
                        END
```

Notice that the addresses start at 0000. The directory entry will look like this:

```
 Size Addr   La   Sa flags Name.
    1 19F8   17    0     N RL.RL
```

The load address contains the start of the word-map and the Start address contains 0. Notice that the above listing stops at byte 16. The word-map is not printed to the listing during the assembly process. To see the word-map, we will need to look at a dump of the file. Here is a partial dump:

```
0000 ¦ 21 07 00 CD 0C 04 C9 0C ¦!.......¦
0008 ¦ 47 6F 6F 64 20 6D 6F 72 ¦Good mor¦
0010 ¦ 6E 69 6E 67 2E 0D 00 C0 ¦ning....¦
0018 ¦ 00 00 00 00 00 00 00 00 ¦........¦
```

Notice that byte 17 is "C0". Let's convert that to binary.

```
 C0H = 11000000B
       ↑  - This first bit says that the
first word in the file must be changed.

 C0H = 11000000B
        ↑ - This second one says to start
with the second byte in the word. The net
effect is to cause the word composed of
byte # 2 and 3 in the file to be changed.
Bytes # 2 and 3 comprise the word which has
the value of the label Signon.
```

If the second byte were 0, then the change would start with the first byte in the word. To show this effect, I have added a NOP to the file. The source file now looks like:

```
        REFS SYSTEM.SY
        REF USER
        REF Msg
        RELOC
        NOP
Start   LXI H,Signon    ;Point HL at our message
        CALL Msg        ;Display message pointed to by HL
        RET             ;Back to Exec
Signon  DB 0CH,'Good morning.',0DH,0
        END
```

The output of the assembler looks like this:

```
0000  00              NOP
0001  210800    Start   LXI H,Signon    ;Point HL at our mess
0004  CD0C04            CALL Msg        ;Display message poin
0007  C9                RET             ;Back to Exec
0008  0C476F6F  Signon  DB 0CH,'Good morning.',0DH,0
000C  64206D6F
0010  726E696E
0014  672E0D00
```

And the directory listing looks like:

```
 Size Addr   La   Sa flags Name.
    1 19F9   18    0     N R1.RL
```

Again, to see the word-map, we need to examine a dump.

```
0000 ¦ 00 21 08 00 CD 0C 04 C9 ¦.!.....¦
0008 ¦ 0C 47 6F 6F 64 20 6D 6F ¦.Good mo¦
0010 ¦ 72 6E 69 6E 67 2E 0D 00 ¦rning...¦
0018 ¦ 40 00 00 00 00 00 00 00 ¦@.......¦
```

Because of the added byte, the word-map starts at byte 18 with a 40H.

```
 40H = 01000000B
       ↑ - This bit says do not change the first word.
 40H = 01000000B
        ↑ - This bit says change the second word.
 40H = 01000000B
         ↑ - This bit says start with the first byte.
```

The first word is 00 21 (the NOP and the LXI H,). The second word is 08 00 and is the relative address of the string FF,'Good morning.',CR,0 in the file.

Now that you know everything there is to know about the word-map, you can load relocatable files helter-skelter anywhere in the 8080 address space, right? Well, here's how it works.

First we need to read the file into memory at the location where it is going to be executed. Then we need to read the word-map and figure out what addresses in the file must be changed. Then we must add the actual load address to the word to be changed in the file.

In the first file above, suppose we were to load the program at address 3200H. The word-map of C0 says that the first word in the file, second byte (07 00) would need to have 3200H added to it. Remembering that the 8080 stores words in byte-reversed order, we must convert the "07 00" 0007 and add that to 3200. We get 3207. When we put it back into the proper location we would convert it back to "07 32" So, our dump of MEMORY now would look like this:

```
3200 ¦ 21 07 32 CD 0C 04 C9 0C ¦!.....¦
3208 ¦ 47 6F 6F 64 20 6D 6F 72 ¦Good mor¦
3210 ¦ 6E 69 6E 67 2E 0D 00 C0 ¦ning....¦
3218 ¦ 00 00 00 00 00 00 00 00 ¦........¦
```

Notice that, except for the left over word-map, this is now exactly the same as our file dump of the original non-relocatable program which ORGed at 3200. We can now start the program at 3200 and it will run just like the original. Since the word-map is past the end of the program, it will have no effect.

When it comes to relocating a program up under MEMTOP, a slightly different strategy is required. If we were to load the program under MEMTOP by the number of sectors in the file we would be loading the word-map as well as the program. We would have stolen valuable memory that would be unused. On the other hand, we can't use Dio to load the program directly where it would end up, because the word-map would write on top of stuff above MEMTOP. Either there would be no memory there, in which case the word-map would be lost, or there would be a TSR already resident, in which case it would get clobbered!

A strategy for overcoming both these limitation would be to load the relocatable file into user memory, change the necessary addresses, and then move only the program and its buffers up under memtop. In the case of the above program (the one without the NOP), only the first 16H bytes would need to be moved. (Byte 17 is the start of the word-map).

Writing software to be relocatable is mostly the same as writing non-relocatable software. But, there are some differences. One can't use ORG to find a sector boundary. Also, one must be very careful with self-changing code. But, that is enough for another article.

1989 Annual Index

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 UNDELETE

        HELP file for system command "UNDELETE"

The "UNDELETE" command undeletes all deleted files in a
        directory.

Syntax: "UNDELETE [<n<path<file.DX]" (RETURN)
'n' is a drive number and file is a directory file.

"UNDELETE " undeletes files on the system resident drive.
"UNDELETE [n]" undeletes files on drive 'n'.
```

"UNDELETE [<n<path]" undeletes files on drive 'n' and in subdirectory path.

Minimum size: "UN" Example "UN <2<LETTERS.DX"

HELP PROGRAM DIRCOPY

HELP file for system program DIRCOPY

This utility copies all files within a directory to another directory, and all of its subdirectories. It is invoked as follows:

For main directories on unit 4 to unit 5:

DIRCOPY 4 5 *

is essentially an IMAGE and PACK in one operation.

For SubDirectories Sub1 to Sub2:

DIRCOPY <d<Sub1 <d<Sub2 *

where the optional "*" means replace the file if presently in the directory, "d" is the option disk unit number. Many combinations are possible, i.e., copying from a main to subdirectory, from two subdirectories, etc.

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.

If the primary function being performed is a backup of a complete disk rather than a general subdirectory copy, the program BACKUP supplied on the system disk should be used. It checks the "New" bit of each file and resets it after a successful copy, thus not repeating the copying process with the first file of the directory endlessly. BACKUP otherwise operates identically to DIRCOPY.

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. Noncommercial ads by subscribers are free.

Entire 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.

DISKS - MODEMS - PROMS - SOFTWARE - SPELL

1. MAXELL diskettes: 5-1/4" 10 hard sector -- $15 per box.
2. Used diskettes: 5-1/4" 10 hard sector -- $0.50 each.
3. Hayes Micromodem 100 (300 baud S-100 internal modem) $30.
4. HayesSys modem software (for the Micromodem 100) $30.
5. Abstract Systems Exec (Enhancements & bugs corrected) $35.
6. Abstract Systems Proms (Enhancements & bugs corrected) $35.
7. PolyGlot Library Volumes: $6 each; 5 or more - $5 each.
8. Hayes Smartmodem 12008 (IBM compatible internal) $60.

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

Remember last issue's Bit Bucket? Well, the outcome was a binary dis-stink-shun.

For Beginners

Most beginners are often perplexed when confronted with a computer, especially the first time. "What do I do now?" is the question uppermost in their minds. "How do I get it to do what I want it to do?" Learning to use a computer has some difficulties not found in learning to do other things. A failure to do things right in other contexts usually gives some information which will enable the learner to do something better. With a computer an incorrect input often gives no information.

Computers are designed to do only a few basic things. To learn to use a computer, one must learn what those things are it can do, and how to tell it to do them. HELP systems and manuals tell you what these possibilities are. The rest is figuring out ways to combine the things it can do to produce something like what one wants.

One way to make learning the initia possibilities easier is to install a menu system. This allows presenting the options to the user so he or she can choose the desired one with a minimum of fuss.

In This Issue

Coming Soon

Poly Meta, Managing TSR programs, More BASIC for Beginners, System Programmers Notes, Help, BugNotes, Public Domain Software, etc.

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

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 ads are free of charge. PolyLetter is not affiliated with PolyMorphic Systems.

Back volumes of *PolyLetter* (1980 thru 1988) are available at reduced prices payable in US dollars to Ralph Kenyon. 1 - $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 ($3.50, $4.00, $5.00).