PL/I-80 from Digital Research

Big Machine Language Meets Micro

By Tim Barry

PL/I. The very name of this monster language conjures up images of massive mainframes and giant OS/MVT operating systems. The introduction of PL/I-80 indicates that Digital Research, the originator of CP/M believes microcomputer software developers are ready to add large-system capability to their home and small-business systems.

PL/I (pronounced PL-One) stands for Programming Language One—as in the only one—as in the only programming language you will ever need.

IBM developed the language in the mid-60s as a replacement for FORTRAN, ALGOL, COBOL, assembly language, and nearly everything else then in use. That all these languages still exist and are, with the exception of ALGOL, more widely used than PL/I shows the one major flaw in the master plan: to be all things to all people requires a BIG language.

PL/I was so large, general, and complicated that PL/I programmers required a lot of training. Also, around this time, the basic laws of software inertia became obvious, to wit: "I don't care if it's written in Latin; the program works and we're going to leave it alone."

The early languages, despite their limitations, were entrenched by the cost of re-doing existing programs. The net result was that PL/I never received the wide-based acceptance that its supporters (myself included) felt it deserved.

The microcomputer software scene today is strangely reminiscent of the large-system scene in the mid-60s. The entrenched languages this time are BASIC, FORTRAN, and Pascal; and once again, PL/I is late arriving on the scene. I hope it's not too late this time, because the microcomputer community will lose if PL/I-80 doesn't make significant market penetration.

I contacted the Digital Research People about doing the program review and received total cooperation. They sent their current license agreement: PL/I-80 buyers must be CP/M licensees, and the license agreement must be completed and returned to Digital Research before the compiler is shipped. I completed and returned the agreement, and received my review copy less than one week later.

The PL/I-80 package comes on two standard single-density 8-inch diskettes and includes over 320K of code in various types of files. The written documentation consists of three 8½ x 11 manuals, totaling 309 pages, and a 28-page pocket PL/I reference guide. The entire package costs $500; separate manual sets are $35.

The compiler package also includes 46 sample PL/I programs; a library manager; a linking loader; and RMAC, a relocating version of the Digital Research MAC macro-assembler. These last three plug a large gap in the existing line of Digital Research software development tools. I hope they will be offered for sale separately.

Functionality

Covering the functionality of 150K of systems programs in a newspaper review takes a long time. I'll have to be content with an overview of the system and some comments on the few problem areas that turned up.

PL/I-80 is a three-pass compiler that produces relocatable object records. These records are then "link-edited" to incorporate any required programs from the run-time library, and everything is fixed to run in the target hardware configuration. The default configuration is a CP/M-compatible direct-execution core image ("COM" file), but the user has complete control over location of both code and data segments. This makes the compiler suitable for generating ROM-based code for dedicated applications.

The library manager stores object records generated by both the compiler and the RMAC assembler in user program libraries. Most users can cut program development time by storing commonly used routines in libraries and then linking them to new programs under development.

The linkage editor generates an adjusted symbol table containing the actual load address of all the labels and subroutines linked into the final load module. This allows use of the symbolic debugger (SID, also available from Digital Research) to debug the execution of compiled programs.

PL/I's design corresponds closely to the ANS PL/I Subset G language definition. Digital Research whacked out the same large hunks of PL/I as a lot of other users. This must be done to make a language the size of PL/I practical on a small computer. The subset chosen should have no impact on most users; indeed, this compiler represents the most effective higher-level language yet available to microcomputer users.

The compiler runs in a normal CP/M environment with a minimum of 48K read/write memory. I compiled programs ranging from 15 to 730 lines with no problems. While the compiler won't win any speed contests, it is fast for doing a fully optimizing pass to reduce the size of the generated code. On my three MHz 8085 system with Shugart SA-800 8-inch floppy disks, a usable speed estimate is about 300-500 lines per minute for small to medium programs...somewhat slower for

InfoWorld

Software Report Card

PL/I-80
Digital Research

Functionality	***Excellent***
Ease of Use	***Excellent***
Documentation	***Good***
Error Handling	***Excellent***
Support	***Excellent***

System Requirements

- CP/M Operating System
- 48K RAM

Price: $500
Digital Research
Box 579
Pacific Grove, CA 93950

larger programs. I don't expect users to compile many large pieces of in-line code; it's more efficient to break the program down into modules and compile them separately.

Reasonably efficient code comes from the compiler. A 19-line program to compute and print integer factorials compiles to a 165-byte relocatable object file. When linked to form a load module that can run, the final COM file size is 7.5K, indicating a run-time overhead of about 7K. That is not unreasonable given the thoroughness of the run-time error-handling system. Overhead is comparable to other compilers'. Execution is fast.

One limitation on the utility of the review version (Release 1.0) of the compiler is its lack of program chaining or run-time overlaying. This limits the size of the largest program to the size of available system memory. Since all compilers, even optimizing ones, are memory-hungry, this can be a problem for users developing large, menu-driven program systems. Digital Research says that Release 1.1 of the compiler and linker will support full overlay structures. They should be available by the time this review is in print.

Ease of Use

Since PL/I-80 is a pure compiler, it requires a more disciplined development approach than many programmers use. Unlike BASIC, where errors can be corrected immediately and the program re-run, a compiler's source programs must be edited, compiled, link-edited, and then run every time a source change is required. This puts a premium on program-logic and program-entry precision. The compiler has great capabilities, but it is like any other sharp tool: you have to use it carefully.

Control of the compiler, library manager, and link editor is easy. All controls are direct and easy to learn, and the default values chosen make typing two words sufficient to compile most programs. A variety of toggles is also available to control the compilation process.

Documentation

The 300-page documentation package consists of a 98-page language manual, a 134-page application guide, and a 77-page LINK-80 operator's guide. These well-written manuals provide numerous examples in the form of actual programs supplied on the sample diskette.

As good as the documentation is (the manuals have a complete table of contents, for example), room exists for improvement. I feel there should be a fourth main manual—a PL/I-80 compiler operator's manual. It should gather up all the sections concerning compiler invocation, control toggles, pass-1 errors, pass-2 errors, pass-3 errors, and run-time errors; expand them; and combine them with examples of start-to-finish output.

The reference summary covers much of this information in a form suitable for experienced users, but a more centralized and detailed presentation of this material would help new users. The current documentation does not provide enough information about various types of compile and run-time errors. (See next section.) It also doesn't have an index, an indispensible item.

I'm not being picky; I appreciate Digital Research's program. It is difficult to adequately define a programming language in a way that suits the requirements of all users. The application guide and example programs represent a thorough "hands on" course in PL/I programming. The interfacing guide in the LINK manual is the best example of a higher-level language/assembly-language interface I've seen.

Error Handling

PL/I-80 has thorough compile-time error analysis and a flexible run-time error-handling system. The relevant information is spread throughout the manual set, though, so I had to hunt to find it all.

The compiler does a fine job of finding errors and displaying them on the screen. It displays the offending line, shows the character position, and prints the error message. What's needed are more error examples showing *why* the statement you enter, which looks fine, is, in fact, a syntax error. This is crucial for new users who are not used to compilers. On the plus side, the documentation contains a fine section on exception processing.

Support

Digital Research developed CP/M; Digital Research developed PL/I-80. PL/I-80 will be well supported.

Summary

PL/I-80 is a bold attempt to bring a much-needed tool to the microcomputer software community. Gary Kildall, president of Digital Research; designer of PL/M, CP/M, and PL/I-80, is to be commended for taking the hard road, when a FORTRAN, Pascal, or BASIC compiler would have been an easier commercial success. On the whole, the compiler is a great technical achievement. Users willing to undergo the momentary discomfort of parting with old, familiar tools will be rewarded by PL/I-80. It is a capable, flexible language whose few deficiencies are sure to be corrected with time. I hope it's not too late. ■

"Reprinted with permission from InfoWorld, Vol 2, #15 September 1, 1980. 530 Lytton Av., Palo Alto, Ca. 94301. $18/yr US; $35/yr Canada and Mexico; $65/yr other foreign."