The PL/I Connection

Issue 4 **IBM Santa Teresa Laboratory** **September, 1995**

Editor: Shauntel Christensen

PL/I for AIX? -- You Bet!

In keeping with a continued commitment to provide cross-platform solutions to customers, IBM has announced that its robust programming language, PL/I, is now available on the UNIX platform.

PL/I Set for AIX allows you to take advantage of expertise you already have invested in PL/I while developing new client/server applications on your workstation. You can also port existing mainframe applications or those created with PL/I for OS/2 to the AIX environment.

Here are just a few of the things you can do with PL/I Set for AIX:

- Access subsystems like DB2 and CICS.
- Take advantage of tools common to other AIX programming languages. For example, the Program Builder manages the tasks of compiling and correcting source files and linking the resulting object files into an executable program or library. The Program Debugger provides a windowed environment that helps you detect and diagnose errors.
- Benefit from many of the same language features you find in PL/I for OS/2 including new built-in functions, strongly typed enumerations, a variety of storage classes, restricted expressions, and named constants.
- Preprocess code using a state-of-the-art macro facility.
- Facilitate moving applications from the mainframe to the desktop by using data utilities included in the product. In addition to support for local and remote access to DB2, CICS, and VSAM, you get host emulation for EBCDIC and hexadecimal floating-point data.

IBM PL/I Set for AIX Version 1 is available on CD-ROM or 8mm tape. Its generated object programs run on RISC System/6000 family processors with IBM AIX Version 4.1.3 (or subsequent releases).

To order PL/I Set for AIX, contact an IBM representative or from the following countries, please call the corresponding number:

Austria	0222 21145 2500
Canada	1 800 565 SW4U
France	36 63 36 43
Germany	0130 4567
Italy	16 70 17 001
Netherlands	06 0220402
Switzerland	155 1225
UK	01329 242728
USA	1-800-IBM-3333

Please ask for part number 33H1858 (CD-ROM) or 33H5425 (8mm tape).

And now...PL/I on the Web

The PL/I home page is available on the World Wide Web. You can follow a path to the PL/I home page by first going to the IBM software home page at URL **http://www.software.ibm.com** and bringing up the "Software Index". Under the letter "P" in the index, you can select *PL/I Family*. Once you get to the PL/I home page, you will have a repository of PL/I information at your fingertips, including the history of PL/I and the latest and greatest product announcements.

By using the path described, you can take advantage of the wealth of information about IBM product offerings that is available on the software home page. If, however, you would rather skip the software home page and go directly to the PL/I home page, use URL **http://www.software.ibm.com/ap/pli/plihome.html**.

PL/I for OS/2 Price Reduction

During the last week of September, IBM announced a 40% price reduction on PL/I for OS/2 Professional Edition. The price for both the PL/I for OS/2

Personal Edition and PL/I for OS/2 Toolkit have been reduced as part of the same announcement. Current prices (in U.S. dollars) for PL/I for OS/2 products are as follows:

Professional	$749
Personal	$229
Toolkit	$149

A copy of the complete announcement letter is accessible from the PL/I home page.

Going to GUIDE?

GUIDE Fall '95 is scheduled from November 5 - 9 at the Sheraton New Orleans. PL/I sessions include the following:

AP201 – Condition Handling in Application Development (PL/I Focus)

Condition handling paradigms of three common development languages (PL/I, C, and COBOL) will be outlined and discussed. Merits of the powerful PL/I condition handling model will be identified with examples. Attendees will be encouraged to talk about their current use of condition handling and what they would like to see in future offerings.

Speakers	Don Smith - IBM
	Peter Elderon - IBM
Moderator	Sue Bortone - NYNEX
Wednesday	8:30 a.m.

AP202 – Porting PL/I from OS/2 to AIX

Peter Elderon, the chief developer for IBM's PL/I on the workstation, will describe his experiences porting a non-trivial PL/I application from OS/2 to AIX, namely the PL/I compiler and library itself. This talk will concentrate on this user's experience, and include tips for those of you who want to port PL/I from either the host or OS/2 to AIX. (It will also describe why the few bits written in C were the least portable.)

Speaker	Peter Elderon - IBM
Moderator	Sue Bortone - NYNEX
Wednesday	10:15 a.m.

AP203 – PL/I Requirements (An informal overview with IBM at SKIDS)

Grab something cool to drink, a handful of munchies, and mosey over to the PLIwood table at SKIDS to find out what happened to that PL/I requirement you submitted. IBM has updated the database and will have all the latest on status. If you're interested in checking up on a past requirement, submitting a new one, or just want to chat about things you'd like to see in PL/I on any platform, this is the place to be. Drinks are on us!

Speakers	Roz Palmer - IBM
	Karen Barney - IBM
Moderator	Sue Bortone - NYNEX
Wednesday	6:30 p.m.

Besides these specific sessions, other PL/I related sessions--for example, a Languages User Exchange (AP049), and presentations on IBM's Language Environment (AP101, AP102, and AP104)--are scheduled on Thursday. Hope to see you all there!

Table of Contents

To 'C' or Not to 'C' Is Recoding a Legacy PL/I Application to C The Right Thing to Do?

by Richard Perkinson, Liant Software Corporation
dickp@lpi.liant.com

This is the third of four articles in a series which began in the March issue of "The PL/I Connection."

Part 3 - Recoding from PL/I to C: The Specific Concerns

PL/I is a particularly rich language, and many of its features cannot be translated simply into C. The conversion is possible, but you will have to expend a lot of effort. The following is a list of PL/I features and characteristics that will need special attention if you choose to recode your PL/I application to C.

Data Support

The following PL/I data types and classes have no direct equivalent in C:

- Fixed Decimal(p,q), or "packed decimal"
- Pictured numeric, things like PICTURE 'S(5)9V99'; you use these in arithmetic operations
- Picture edited, such as PICTURE '(6)$9V.99-'
- Fixed-length character strings must be handled as array of CHAR in C; various complications are associated with character strings, such as the need for null terminators in C, the space padding in PL/I, and the failure to truncate in C
- Bit strings
- Unaligned Fixed Binary and Float Binary
- Defined with the Position attribute
- Automatic arrays with variable bounds and strings with variable lengths
- Structures using the Refer option
- Area and data offsets
- Controlled storage
- Default statement
- Character Varying

In addition, data initialization in PL/I is quite flexible and some of it will require special attention when replacing the functionality with C code, for example:

- String repetition factors
- Array initialization
- Initialization of automatic and based data
- Initialization with expressions, including functions
- Assignment of the null string to a structure, which sets all character members to space and other members to zero

The lack of sufficient data type expressions to fully define the many PL/I supported data types is the most troublesome aspect of converting to C as discussed in the section entitled *"Automatic Code Translators"* earlier in this paper.

Data Operations

Some of the kinds of operations you can do in PL/I will require special attention [when converting PL/I to C]:

- Array operations
- Array cross sections
- Structure operations
- Structure operations "by name"
- PL/I "pseudo-variables", such as Substr and Unspec
- PL/I's implicit type conversions
- PL/I type conversion rules, which are quite distinctive
- Get and Put string

Control Structures

The following aspects of program logic are different enough in PL/I that they will require special attention:

- Nested procedures and begin blocks; these also affect the scope of names
- Multiple entry points
- Parameter passing, which is usually by reference in PL/I
- Variable argument lists
- PL/I's extensive condition handling, including user-defined conditions

File I/O

The translation of PL/I's I/O will, of course, require special attention. In particular, those [I/O types] needing attention are:

- Sequential record I/O
- Indexed record I/O
- Relative record I/O

- List-directed stream I/O, with its automatic data conversion rules
- Get and Put data operations
- Many of the special features of PL/I edit-directed I/O, such as column specifiers and variable field widths
- Certain of PL/I's format specifiers, such as A, B, COL, P, R, LINE, PAGE

Built-in Functions

Some of PL/I's built-in functions will require special attention, for example:

- All and Any
- Bool
- Ceil and Floor
- Collate
- Prod and Sum
- Round and Trunc
- Substr
- Translate
- Valid
- Verify

Macro Preprocessor

Some of the features of the PL/I Macro Preprocessor will require special attention, for example:

- %Activate and %Deactivate, including the use of Rescan and NoRescan
- %Assign
- %Procedure

Sample Recoding Consideration

A simple example of the care that must be taken occurs in a case like this:

```
declare NotOftenUsed String char (14);
declare BusyString char (16);
NotOftenUsed String = BusyString;
```

Perhaps, over the years, someone found it necessary to increase the size of `BusyString` without noticing that it was somewhere assigned to `NotOftenUsedString`. It never mattered, though, because no one seemed to notice that two bytes were cut off the end of `NotOftenUsedString` in this case. However, when this assignment is converted to a strncpy call in C, two bytes of memory get overwritten. The bug might surface right away, but most likely, it will take real users to hit it.

To receive this paper in its entirety, please call 1-800-818-4PLI ext. 221 or (508) 872-8700 ext. 221, or send email to openpl1@lpi.liant.com.

PL/I Provides Connectivity Options to TAB

by Neale Ferguson, Totalizator Agency Board of New South Wales, Australia
neale@gnome.tabnsw.com.au

The Totalizator Agency Board of New South Wales (TAB) is a statutory authority charged with the responsibility of providing off-course wagering facilities for the adult population of the state. Last financial year, the TAB turned over $A3.5 billion on wagering on galloping, harness racing, greyhound racing, and sporting events.

The TAB has a large investment in a private SNA network as well as providing some access via X.25 public networks. With the rapid increase in interest in TCP/IP in general and the Internet in particular, there was a perceived need to examine the commercial, security, and technical issues of this technology.

A pilot project was commenced, the purpose of which was to enable access from a TCP/IP based network to our VM and VSE based host systems. The application was to act as a protocol converter. The wagering systems resident on VSE/ESA primarily talk to wagering devices via LU 0 protocols. Thus, the function of our protocol converter was to enable sockets applications to speak the lingo of LU 0.

Concurrent with this was the acquisition of PL/I [for OS/2] Professional Edition. The TAB has a long history of using PL/I for both VM and VSE in its production systems. Therefore, the ability to leverage its investment in personnel and their skills is attractive to us. We consequently made the decision to build the prototype using the tools of PL/I for OS/2.

Very generally, the project consisted of the following components:

1. Developing skills using the PL/I product including the visual [PL/I for OS/2] Toolkit.
2. Analyzing the requirements of the protocol converter and translating this into design specifications.

3.Conversion of TCP/IP 'C' header files to PL/I include books.
4.Conversion of CM/2 LUA header files to PL/I include books.
5.Building a message repository.
6.Defining and generating DB2/VM tables which contained control information regarding trusted users, SNA logical units, and TAB authorized applications. OS/2 based applications that extracted this data via DDCS/2 and built the controlling in-core tables were subsequently designed and coded.
7.Coding and testing of the prototype.

The visual toolkit was primarily utilized to provide the protocol converter with a logging facility rather than using VIO based windows. For the rest of the 15 or so modules, standard OS/2 editing facilities were used to build the code. The multitasking facilities of PL/I were to be used extensively.

The C2PLI utility was used extensively to convert the TCP/IP and LUA header files into their PL/I equivalents. There was some significant hand tailoring of these generated files for aesthetic and technical reasons. The author almost died of fright when these files were inadvertently erased (just prior to installing an ADSM client on my machine!) After recovering from this disaster, work continued normally.

The build/make and make facilities of Workframe were utilized to generate the application from the source code. Workframe is an extremely useful and usable feature of the PL/I offering. Once I got my mind around all of the various facilities (access profiles, build, make, etc.), work progressed smoothly.

After only about seven days of effort, the application was ready for testing. Generally, problems were caused by:

1.Unfamiliarity with calling techniques (for example, pass by address is not a good option when attaching subtasks).
2.Incorrectly converted header file statements (generally due to incorrect manual reworking of C2PLI output).
3.Implementation of the LU 0 protocol (for example, bracketing).

Testing was assisted by facilities such as subscriptrange checking and use of the PLITEST facility. PLITEST cannot be used in a PM environment with 100% success. I hope that in the future, enhancements will be made to allow this. In a similar vein, enhancements to PLIDUMP to make it more like its S/390 cousins would be useful. Of concern is the inability to dump DSA and TCA storage areas.

Following two weeks of unit testing, a test was conducted that used an Apple Newton to communicate with the protocol converter over a private mobile TCP/IP network. Within a couple of days, the Newton was able to access our online test systems and generate wagering transactions. The protocol converter worked reliably and was not resource greedy.

All parties involved in the project were pleased with how quickly the application was developed and the standard of the code produced. I believe the lack of problems encountered can be ascribed to the following:

1.A reliable and robust PL/I compiler.
2.Utilizing the existing PL/I skills of a seasoned programmer.
3.Productivity tools such as the visual toolkit, C2PLI utility, and build facilities.
4.Useful debugging aids such as range checking and PLITEST.
5.PL/I's implementation of OS/2 multitasking.

We now keenly await the porting of this PL/I compiler to the S/390 platform.

Peter's Performance Tips

by Peter Elderon, IBM PL/I Development
elderon@vnet.ibm.com

In the last issue, I discussed some of the new features of PL/I for OS/2 and how they could be used to help improve the performance of your code. The first half of this set of tips continues that discussion, while the second half discusses an improvement in performance that resulted from a recent improvement to our compatibility with the host compiler.

When we introduced ordinals, we also expanded the varieties of DO loops by allowing UPTHRU and DOWNTHRU in type-3 DO loop specifications. While these keywords are needed to write DO loops with ordinal control variables and DO loops where the ending value is the last value the control variable could assume (example: 32767 for a FIXED BIN(15) control variable), these keywords can also allow you to write code that performs slightly better than a normal DO loop.

For example, many DO loops have the form:

```
 do i = n to m;
     ...
   end;
```

If you know that n is at least as big as m, you also write this loop as follows:

```
   do i = n upthru m;
     ...
   end;
```

When UPTHRU is specified in the DO loop, a test is not made before the first iteration of the loop. Hence, this form of the loop performs better. Moreover, the compiler also optimizes this form of the loop better.

In the loops above, the code also performs better on OS/2 if the loop control variable is FIXED BIN(31) rather than FIXED BIN(15) or FIXED BIN(7). This is one of the reasons why a statement in the performance chapter in the Programming Guide is not quite true: the choice of ANS or IBM as a suboption of the DEFAULT compiler option does have a small effect on performance. (The Programming Guide claims that it does not.)

Specifying DEFAULT(ANS) makes your code perform better if you have incomplete declares. Under DEFAULT(ANS):

- Variables declared only as FIXED BIN default to FIXED BIN(31), and code using them generally performs better than code with variables defaulting to FIXED BIN(15).
- Variables declared without any attributes default to FIXED BIN(31), and code using them generally performs better than code containing variables with names beginning with letters other than I-N (which default to FLOAT).

Of course, specifying DEFAULT(IBM) assures better compatibility with the mainframe which can be very important in getting the same results on the workstation and host. (Not to mention that maybe the original programmer of some of your old code knew that "DCL F;" would make F have the attributes FLOAT.) Because we know how important cross-platform compatibility is to you, we put this option (and many other features) in the original version of the workstation compiler.

But, we didn't stop there. Through the Corrective Service Diskettes (CSD's), we are continuing to improve compatibility with the host, and CSD#5 contains some enhancements that not only improve compatibility, but also improve your code's performance.

A short time ago, I saw a mainframe program that contained a test to see if all of the elements of an array (x) were zero. The program made this test as follows:

```
if sum( x ¬= 0 ) = 0 then
  /* all elements are 0 */ ;
else
  /* some elements are not 0 */ ;
```

The workstation compiler refused to compile this statement. Some would say this is good: it would be better to write the test as:

```
if all( x = 0 ) = 0 then
  /* all elements are 0 */ ;
else
  /* some elements are not 0 */ ;
```

It would be better to write this test using the ALL built-in function primarily because it is easier to understand what it is trying to do. Moreover, it also performs better.

- The test using ALL tests each element of the array and stops as soon as a non-zero value is found.
- The test using SUM tests every element of the array (even if the first element is non-zero) and performs an add for every element of the array.

The workstation compiler did not reject this statement because it was poorly coded; the compiler rejected the statement because it did not allow array expressions as arguments to the SUM and PROD built-in functions. This incompatibility with the host compiler caused problems for some of the PL/I for AIX beta customers (who were applying SUM to array expressions in CAD-CAM applications) and was removed. With CSD#5, this restriction is removed from PL/I for OS/2 as well.

In addition to lifting this restriction, the evaluation of all SUM and PROD built-in function references is now done inline. If you want to count how many elements of an array are zero (rather than just determine if any of them are), you could use the expression "SUM(x = 0)", and the compiler generates code to evaluate that expression with an efficient inline loop.

In summary, you get better compatibility and better performance. Who says you can't have your cake and eat it too?

Upcoming Events

There are a couple of events before the end of the year that will feature PL/I demos or presentations. We'd love to have you come by and see first hand what we're up to these days. As was mentioned in a previous article, GUIDE is in New Orleans from November 5-9 and the DataBase/Client Server tradeshow is from December 5-7 in Chicago.

Fax Information Service

Did you know that IBM has a "Fax on Demand" service that allows you to easily retrieve documents for immediate delivery to any fax machine? There is information available on all sorts of IBM and IBM Business Partner products, including PL/I. The service is available in the USA 24 hours a day, 7 days a week. Voice prompts navigate you through the document selection process. To use the IBM Fax service, call: 1-800-IBM-4FAX (1-800-426-4329).

Probably the best way to start is to order the index, and then pick out specific documents from there.

New PL/I Textbook

A new book, Introduction to PL/I, Algorithms, and Structured Programming, is now available. Written by R.A. Vowels (*rav@cs.rmit.edu.au*), this book has a place in the fields of computer science, mathematics, engineering, and science. The following description has been summarized from the material on the back cover of the book.

Introduction to PL/I, Algorithms, and Structured Programming emphasizes fundamentals of structured programming through study of PL/I. It is designed for a reader's first or second exposure to computer programming, and is intended to provide a sound grounding for the reader who desires to study PL/I in greater depth.

The book is organized into two parts. The first part (Chapters 0 to 9) is concerned with elements of PL/I. The second part (Chapters 10 to 24) introduces traditional and new algorithms. The book will be found helpful as a reference. Appendices contain a summary of PL/I statements, a list of built-in functions, and algorithms for some of the new built-in functions.

Date of Publication:	August 1995
Description:	555 pages, soft cover
Cost:	$30 U.S. + post
	$40 Aust + post

Available through mail order from the publisher: Vowels, 93 Park Drive, Parkville 3052, Victoria, Australia.

CSD#5 Now Available

CSD#5 is now available for the Professional and Personal Editions of PL/I for OS/2. Support for BTRIEVE is just one of the new additions to an already powerful product.

Instructions for obtaining CSD's are on page one of the March, 1995 issue of *"The PL/I Connection"*. There has been one slight change to those instructions regarding Anonymous FTP. The CSD's are available on **ftp.software.ibm.com** in the **/ps/products/pli/fixes** directory.

Tell your Friends about PL/I

In our last issue, we told you about the IBM Redbook, titled *"PL/I for OS/2; PL/I for OS/2 Toolkit: Visual PL/I; CODE/370 PL/I Support"*. We just wanted to remind you that it is still available and the publication number is GG24-2501-00. If you have any questions about Redbooks, you can send a note to redbook@vnet.ibm.com.

If you know someone who uses PL/I and might be interested in receiving this newsletter, tell them about us! Have them send their name, address, phone and fax numbers, and electronic mail address to one of the following addresses:

PL/I Newsletter
IBM - Santa Teresa Lab
555 Bailey Avenue, B3T/D284
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Fax (408) 463-4820
teampli@vnet.ibm.com
USIB5RLG at IBMMAIL

Questions and Answers

Question	Answer
I am currently evaluating PL/I for OS/2, both the Personal and Professional Edition. We have a very old legacy system (20+ years) written in mainframe PL/I. Thus far, I have been able to compile our programs accross the two platforms. We are just in the process of setting up a client server environment. 1) Can we install PL/I for OS/2 in this environment, or do we need one compiler per workstation? 2) We will have VSAM files sitting in the mainframe (which is not our server). What is needed to access these files? We are going to access the mainframe via Communications Manager/2.	1) One license is required for each workstation from which the compiler is invoked. This does not apply to run-time DLLs. If two programmers are working on a PL/I application to be distributed, you would need two compiler licenses, one for each workstation from which the compiler is invoked. The run-time DLLs can be distributed with the application at no cost. 2) For remote file access, you need the following products in addition to PL/I for OS/2: • On the mainframe side, you need to have MVS/ESA 4.3 or 5.1 and DFSMS/MVS 1.2.0. • On the workstation side, you need to have the Distributed File Manager which is orderable as a CD-ROM, SK2T-8706.
We're writing APIs in PL/I that, in theory, can be called from any other language. We know that they will be called from other PL/I programs and will probably be called from C or C++ programs as well. Since these APIs will accept and return parameter values, it raises the question: Is there a C or C++ equivalent of the FIXED DEC type in PL/I?	370 C has an equivalent for FIXED DEC, but OS/2 and AIX C do not. With C++, you could always define a class to mimic packed operations.
Could someone please clarify IBM's position as far as support for PL/I V2R3. Are the compiler and run time still supported?	PL/I V2R3 is still supported.
Can PL/I include Assembler statements?	PL/I for OS/2 cannot include Assembler statements, however, you can always call an Assembler routine.

I have a customer who has PL/I programs compiled and run under PL/I 1.5.1. Can these programs (without recompilation) run using PL/I 2.3 run-time libraries? We have a requirement to upgrade their PL/I run time to 2.3, but the customer does not want to recompile every program.	Provided that the programs are self contained (they don't fetch or load anything else) and provided you neither recompile nor relink them, then yes, they will work in a 2.3 environment. Otherwise, things get more complicated. The PL/I Programming Guide describes the compatibility issues. For an alternative view, including information about some of the more challenging real world situations, I will send you a document I wrote back in 1989 called "Migration to Version 2 of the PL/I Compiler". It was written when the latest release was 2.1, but it is still valid for 2.3. If you have access to the Web, you can read this document (during *my* working hours) at URL: **http://jamesmf.portsmouth.uk.ibm.com/pli2.htm** *Martin James, IBM UK* *martinf_james@uk.ibm.com*
Is there an area in the compilation listing where it tells the total in bytes of the resulting object module? I cannot discern such a "bottom line" number looking over listings -- perhaps I am not looking in the right place or do not have an option set.	On MVS and VM, PLIOPT produces more than one CSECT. The link edit (MVS) or loader (VM) will also include additional "resident" library routines in the executable module. At run time, additional "transient" library routines may be loaded as well. Of course, the executable may use considerable dynamic storage too. I don't know about other platforms (OS/2, AIX, etc.). This is a complex subject, depending on what you mean by "object module". Your best bet may be a link edit listing. The LIST option will show you most everything that is generated.