The PL/I Connection

Issue 1 IBM, Santa Teresa Laboratory December 1994

Welcome!

Welcome to the first edition of "The PL/I Connection", a newsletter for the PL/I development community. This newsletter is produced by IBM PL/I Development, but we don't want to be the only contributors! If you have a tool you developed, a favorite language feature, a performance tip, a tricky problem you solved, or anything you think other PL/I folks would like to hear about, pass it on! This is a place to share experiences and information. We also want you to know that you're not alone - there are lots of other folks out there who love and use PL/I. Hopefully we can help each other do our jobs better. We plan to publish the newsletter quarterly, but that may vary depending on how many articles accumulate. If you have any questions, comments or ideas, please let us know using the addresses on the last page. We hope you enjoy what we've come up with this time!

Travel Reports

GUIDE

by Karen Barney, IBM PL/I Development

The Fall GUIDE meeting was held in Atlanta, Georgia, from November 5 to November 11, 1994. The keynote address, "The Information Highway: Moving Data to the Fast Lane" was given by IBM Senior Vice President, Ellen Hancock. She presented possibilities becoming reality and pointed out that this is more than a vision - it is understanding technology and taking advantage of what it can do for you. The key to unlocking the power of the information highway is software: operating system, network connection, system management, data base, application development, desktop software - the right software. A demo of WARP included the ease with which one can find the on-ramp.

Jim Archer, IBM Division Director of Object Strategy and Implementation, presented IBM's OO Strategy. He discussed objects being shared across language environments (using SOM) and the building of parts catalogues which are used by professional specialists, the end users, to build applications meaningful to their business, using OpenDoc. If you're interested in OO PL/I, please let us know!

There were two presentations specifically for PL/I. The first addressed the removal of several inhibitors of migrating to LE. These changes will be available in the next releases of PL/I for MVS & VM and Language Environment for MVS & VM. **Note the name changes from the xxx/370 form.** (Editor's Note: See the article, "PL/I Migration to LE - Compatibility Support", for more details.) There was also an update on vendor enablement for LE. The advice for those of you who have a migration problem due to packages which are not yet enabled for LE is to let us know and also contact the vendor. IBM is there to help them out, but you have more leverage in getting them to do the enablement.

The second PL/I presentation was a demo of the new Visual PL/I for OS/2, which allows you to "paint" PM windows and controls rather than coding those tedious calls; and the code generated is all PL/I!

Neat stuff. You can easily and quickly develop graphical front ends for existing mainframe applications, get started with Client/Server, or use it to develop OS/2 applications. Visual PL/I comes with quite an assortment of code blocks, all ready to use, and allows you to add your own code blocks to its libraries. If you are interested in seeing some demo's of Visual PL/I for OS/2, we have a video tape that we can send you. For more information, send in the Response Form at the end of the newsletter with a comment to that effect.

COMDEX

by Christie Nieuwsma, IBM PL/I Development

The COMDEX PC trade show was in Las Vegas, Nevada again this year. It ran for 5 days from November 14th to the 18th. An estimated 200,000 people attended! We noticed that it had a very international feeling. I was there demo'ing Visual PL/I for OS/2. All IBM products were being demo'ed in the same area (supposedly the biggest one there). There were two other products being demo'ed at the same booth as PL/I - Visual COBOL and Visual RPG. As you may have guessed, we were focusing on Visual programming languages, particularly ones with which people moving from mainframes to the Client/Server world might already be familiar. We were right next to the OS/2 WARP theater presentation, so we had quite a bit of traffic. It seems that people are starting to think about porting their products to OS/2 and are interested in what tools exist to help them. We had several people come by and say: "What! PL/I is still alive?! I studied that back in college. Wow - look at it!" Hopefully, some of the people we talked to will remember PL/I when they think about future development!

Peter's Performance Tips

by Peter Elderon, IBM PL/I Development
(This will be a regular column).

One theme in this series of performance tips will be that the more information you tell the compiler, the better your program will perform. By following this maxim, as illustrated in the discussion below, I reduced one application's running time from over 30 minutes to under 15 seconds.

This application, and several others that I have recently seen, have been suffering from woefully poor performance - and because of a statement that you might think could only help performance. This statement is a GOTO statement, as in the following example:

```
dcl next      fixed bin;
dcl states(4) label init( first, add, delete, last );

  loop:
   goto states(next);
  first:
    /* ..... */
    goto loop;
  add:
    /* ..... */
    goto loop;
  delete:
    /* ..... */
    goto loop;
  last:
    /* ..... */
    goto loop;
  exit:;
```

The branch in this loop could have been coded as a SELECT statement, but I don't want to discuss here which is the better choice. I want to discuss the consequences of the code in the preceding example.

As it appears in the example, this code performs terribly in the versions of OS/2 PL/I that have been offered to date. The GOTO statement becomes a call to a library routine. That library routine deserves the largest share of the blame for how poorly this code performs, but a simple change allows you to avoid the library call altogether.

To understand this change, it helps to understand why there is a library call at all. The target of the GOTO is an element of a label array. This label array, *states*, is a variable that is initialized upon entry into the block containing the GOTO. However, since the array may be assigned other labels, including labels in other blocks, the compiler pessimistically assumes that the GOTO may be an out-of-block GOTO and hence makes a library call.

You can help the compiler make the simple jump you expect (and also avoid the cost of initializing the array upon every entry to the block) by adding the attribute **static** to the declaration of the label array:

```
dcl states(4) label static init( first, add, delete, last );
```

The compiler assumes that any static label array has a constant set of values and that any GOTO to an element of the array must be a simple, in-block GOTO.

IBM PL/I...AND THE BEAT GOES ON!

by Bob Thimsen, IBM PL/I Planning

IBM PL/I has been rumored to be on the skids...you hear things like "IBM isn't investing in PL/I anymore" or "PL/I can't compete with COBOL or C". We're here to say that IBM does believe in PL/I and is continuing to make significant investments in this major enterprise product! Here are a few examples of what has happened recently and some of what you can expect in the future as a PL/I customer:

- March 1993: IBM announced PL/I support for the Language Environment/370 (LE/370). First delivery of the LE/370 PL/I product named PL/I for MVS and VM was in April 1993. This product allows the customer that has both PL/I and COBOL to use a common development and runtime environment with the added feature of having IBM supported interlanguage communication between PL/I, COBOL, and C. In addition, a common debugger is available for the LE/370 languages (COBOL and C today and PL/I in December 1994).

- May 1994: IBM announced a Visual PL/I for OS/2 product set with a Professional, Personal, and Toolkit version that became available in June/July 1994. This product set introduced a full PL/I on OS/2 with visual capabilities, SQL preprocessor, CICS preprocessor, VSAM support, IMS Client Server/2 support, macro facility, and source level debugging. The Professional and Personal products come with IBM's Workframe/2 Version 2 product as a basis for a complete development environment. The Toolkit provides a state of the art visual programming environment for the PL/I programmer. The drag and drop interface lets you develop Presentation Manager (PM) applications using objects in a format that is expandable as well as being easy to use.

- September 1994: IBM announced PL/I for VSE/ESA with product delivery scheduled for February 1995. The long awaited update to the DOS PL/I product has been announced and promises to extend the life of PL/I applications running in the VSE environment.

- October 1994: IBM announced PL/I for AIX. The product delivery date will be announced in first quarter 1995. This promises to be an important delivery for the PL/I customer because it provides a product on the only major IBM platform that does not have a PL/I product from IBM. This PL/I for AIX product will be compatible with the other IBM PL/I products on the host and OS/2 as well as provide some important features for the RS/6000 customer.

  IBM also announced enhancements to the host PL/I product in October. These enhancements include PLICALLA/PLICALLB support to be delivered in first quarter 1995 along with the follow on release of Language Environment/370 which will contain the PL/I multi-tasking feature for LE/370 customers. In addition, CODE/370 support for PL/I will be delivered in December 1994 with multi-tasking debugging support available in first quarter 1995.

- December 1994: IBM is announcing a host PL/I feature that allows the host PL/I customer to buy, at a monthly charge, PL/I for OS/2-Professional and PL/I for OS/2-Toolkit. This announcement includes the capability to purchase the distributed workstation feature in 10 or 50 User Packs. The workstation product will be shipped on the host product delivery medium.

  In addition, starting December 1994, you can get a PL/I Solutions CD from IBM. This CD contains actual copies of the following products:
    - PL/I for OS/2 - Professional, Personal, Toolkit
    - IMS Client Server/2
    - DB2/2
    - CICS OS/2
    - Data Access Services for OS/2
    - Workstation Interactive Test Tool
    - WITT/PM
    - IBM Softcopy Receiver Tool

  You can browse or try out the products, read about the products, go through a multimedia presentation, or order the products using an electronic key!

IBM is investing millions of dollars in the PL/I language and development environment technology. We firmly believe that customers can reap the benefits of new technology, optimize their skills base, and, at the same time, protect their business assets through the power of native PL/I and the ease of maintainability that PL/I offers.

PL/I Migration to LE - Compatibility Support

by Irene Liu, IBM PL/I Development

More OS PL/I compatibility support will soon become available in PL/I for MVS & VM and Language Environment (LE) for MVS & VM. The following lists some major items of the support:

CODE/370 support

CODE/370 R2 in conjunction with PL/I for MVS & VM R1 and LE for MVS & VM R3 with a PTF provides a debugging capability superior to that found in INSPECT for C and PL/I, and equivalent to that found in PL/I PLITEST. The support includes interactive CICS debugging and ILC debugging with C and COBOL. It also provides cooperative edit/compile/debug with a language sensitive editor, compiler invocation from workstation, a windowed debug interface, language specific help on OS/2, and compilation and execution on the host.

OS PL/I Multitasking facility (MTF) support

The next release of PL/I for MVS & VM and LE for MVS & VM will support OS PL/I MTF by using the POSIX-defined multi-threaded environment. A PL/I task will be built on the top of a POSIX Heavy Weight Thread (HWT). Each POSIX HWT has a MVS TCB underneath it. Therefore, a PL/I task is still an MVS TCB based task. When a PL/I task terminates, the underlying POSIX HWT and MVS TCB go away as well. The hierarchical relationship among PL/I tasks is maintained to ensure OS PL/I MTF semantics.

This support requires MVS/ESA SP V5R1 with OpenEdition R2. The OS PL/I MTF syntax and semantics remain the same but each task in an application may experience a timing difference on task initialization and termination. If such a difference occurs, it is imposed by the underlying POSIX multi-threading services and the different priority algorithm used by MVS/ESA SP V5R1. Instead of relying on the underlying system services and PL/I default tasking hierarchy, it is always true that the application must use the EVENT option or variable and the WAIT statement to control the desired execution sequence among tasks.

OS PL/I PLICALLA entry point support

The next release of LE for MVS & VM will support an OS PL/I PLICALLA main procedure recompiled with PL/I for MVS & VM. A separate dataset or txtlib, SIBMCALL, must be placed before SCEELKED at link time for the recompiled procedure.

OS PL/I PLICALLB entry point support

The next release of LE for MVS & VM will support the OS PL/I PLICALLB application as follows, subject to the general rule of the version/release support for the OS PL/I applications:
- OS PL/I load module
- OS PL/I object module
- OS PL/I main procedure recompiled with PL/I for MVS & VM. A separate dataset or txtlib, SIBMCALL, must be placed before SCEELKED at link time for the recompiled procedure.

OS PL/I object and load module support

The next release of LE for MVS & VM will extend the OS PL/I object and load module support to the following:
- Object module - OS PL/I V1R3.0
  CICS macro language is not supported
- Load module - OS PL/I V1R5.1
  Non-shared library, non-CICS, and non-tasking load module must first be ZAPed by IBMRZAPM or IBMRZAPV.

The language support is at the OS PL/I V2R3 level. Other migration tools are extended accordingly. OS PL/I IBMBEER is ignored.

OS PL/I PLISRTx support

The next release of LE for MVS & VM will no longer require the OS PL/I PLISRTx application to relink. All known problems will be fixed. However, it is still recommended to relink those load modules to:
- Avoid any potential problem that hasn't been uncovered

- Allow the next release of DFSORT to be installed with a LOCALE value other than NONE
- Allow the library routine to replace the 24-bit parameter list with the extended parameter list when it invokes DFSORT

OS PL/I heap storage location support

The next release of LE for MVS & VM will use the same rules that OS PL/I used to allocate the heap storage. The following rules control the heap storage allocated above the 16M line:
- The requestor is in AMODE(31)
- HEAP(,,ANYWHERE) is in effect
- The main procedure is in AMODE(31)

This support allows HEAP(,,ANYWHERE) as an installation default, yet the location of the allocated storage is sensitive to the AMODE of both the requestor and the main procedure.

Team PL/I

Are you tired of hearing that PL/I is dead? You're not alone! We think it's time for people who love and use PL/I to get together and start letting other folks know what a great language it is! You may already have tried to do that, but we at IBM PL/I Development would like to give you some help. We don't care where you work or what PL/I products you use. If you love the language, you're invited to join

Team PL/I.

What can a member of Team PL/I do?

- Talk it up and answer questions about it on bulletin boards
- Give demos at user groups or trade shows
- Teach classes at community colleges or universities
- Write articles for magazines (or our newsletter!)
- Be available as a reference for potential customers or users
- Share tools and code samples
- and so on, and so on...

What will IBM's Team PL/I support do for you?

- Publish a PL/I newsletter (you're reading it right now!)
- Provide demo material for IBM's PL/I products
- Provide information on user groups and trade shows
- Distribute information on PL/I bulletin boards
- Provide information about PL/I tool vendors
- and so on, and so on...

It's easy (and free) to join Team PL/I! Just send us the following information:
1. Your name
2. Your company name
3. Your mailing address
4. Your electronic mail address(es)
5. Your phone and FAX numbers

6. What PL/I product(s) you use

We'd also love to hear your ideas and plans for what you can do to promote PL/I and how we can help. It would also be helpful to know what bulletin boards you know of where PL/I people congregate.

You can send the above information, and any other comments or questions to any of the following:

Internet: TEAMPLI@VNET.IBM.COM
IBMMAIL: USIB5RLG at IBMMAIL
IBM internal network: TEAMPLI at STLVM20
FAX: (408) 463-4820
Mailing Address: Team PL/I
IBM - Santa Teresa Lab.
555 Bailey Avenue, J84/D345
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Wouldn't it be great to hear people saying: "PL/I... that's a language I need to learn!"
Let's make it happen!

The REXX PARSE Statement as a PL/I Preprocessor Macro

by Dave Jones, Team PL/I Member

parse (par's, par~z), (v.t., Gram). (1) to describe (a word or series of words) grammatically, telling the part of speech, syntactic relations, etc. (2) To break apart into constituent components.

For the last several years, the programming language REXX has been gaining popularity in both the mainframe environment (where it was first developed in the early '80s under VM/SP as a replacement for CMS's EXEC and EXEC2 command languages) and in the PC environment. REXX (short for REstructured eXtended eXecutor) is particularly suitable for writing command procedures and user defined macros for applications like text editors because it has some very powerful string and text handling capabilities. Like PL/I, which inspired many of its features, REXX is a general purpose programming language with the usual structured programming constructs - IF, SELECT, DO WHILE, LEAVE, etc. - but it also contains a number of other constructs not found in most other languages. One of these powerful statements is the PARSE instruction.

In a REXX program the data to be parsed is called the source string. The PARSE instruction splits up the data in the source string and assigns pieces of it to variables named in a "template". The "template" is a model that tells PARSE how to split up the source string and to what variables it should assign the split-up pieces. Here's a simple example:

```
name='Williams, Scott'
parse var name last_name ',' first_name
```

The character string "Williams" is assigned to the REXX variable last_name and the string "Scott" to the variable first_name, based on the position of the comma in the input REXX variable "name". The template in this example is "last_name ',' first_name".

Templates can contain "string patterns" (like the one above), "positional patterns" or both. A string pattern matches characters in the source string to indicate where to split it; the string pattern can be either a "literal string pattern" (one or more characters within quotation marks), or a "variable string pattern" (a REXX variable name within parentheses). In the variable pattern case, the content of the REXX variable determines where PARSE will break the source string. Here's another example:

```
comma = ','
type = 'guru'
name = 'Shaw, Mary  --  PL/I guru extraordinaire'
parse var name last_name (comma) first_name . (type) level
```

yields

```
last_name = 'Shaw'
first_name = ' Mary'
level = ' extraordinaire'
```

The period toward the end of the PARSE statement acts as a placeholder or "dummy variable" to hold unwanted parts of the parsed string - in this case the string ' -- PL/I'. It helps avoid the overhead of creating unneeded variables. Note that all blanks between words are not discarded.

Positional patterns are whole numbers that identify the character positions where the source string is to be split. They can be either "absolute" or "relative"; an absolute positional pattern specifies a number that is the absolute character position at which to split the source string, while a relative positional pattern is a signed number that specifies the relative character position at which to split the source. The sign of the number (plus or minus) indicates the direction of movement from the start of the source string (for the first pattern) or from the position of the last match; right for positive, left for negative. For example, an absolute positional template might look like this:

```
parse var input var_1 11 var_2 21 var_3
```

This parse statement would: 1) place characters 1 through 10 (including blank characters) from the source variable "input" into var_1, 2) place characters 11 through 20 into var_2, and 3) put characters 21 through the end of the string into var_3. Relative positional patterns can be used to accomplish the same thing:

```
parse var input var_1 +10 var_2 +10 var_3
```

Here the plus or minus sign indicates movement, to the right or left, respectively, from the start of the string (for the first pattern) or from the position of the last match. The position of the last match is the position of the first character of the last match; so the last two examples have exactly the same effect. It's important to note that only by using positional parameters can the matching operation "back up" to an earlier point in the source string.

If you wish to parse the value of an expression, REXX provides the PARSE VALUE (...) WITH ... form of the PARSE instruction. The expression, contained within the parentheses of the VALUE keyword, is evaluated and the result is parsed using the template following the WITH keyword. In this format our first example would look like this:

```
parse value('Williams, Scott') with  last_name ',' first_name
```

A more complete explanation of the rules and usage of the REXX PARSE instruction can be found in the REXX reference documents for VM/ESA or in most textbooks on the REXX programming language.

If you've read this far, you might be wondering "What does this have to do with PL/I?" or maybe "Hey, this PARSE instruction is really powerful. I wish I could use it in my PL/I applications." What this has to do with PL/I is that a fellow named Eberhard Sturm at the University of Muenster's Computing Center in Germany (sturm@uni-muenster.de) wanted to be able to use this powerful programming construct in his PL/I applications, so he wrote a PL/I macro preprocessor procedure that implements all of the REXX PARSE instruction features we've discussed above. The PL/I macro can be found on the OS2BBS PL/I forum, or you can request a copy of it directly from me (IBMMAIL(USQVHWVH), or, for those of you who are on the InterNet: usqvhwvh@ibmmail.com).

In only 320 lines of preprocessor source code, including comments, Eberhard was able to construct a full implementation of the PARSE statement, as far as it is applicable to the PL/I environment. The only difference to the REXX PARSE statement is that items in the templates are separated by commas instead of blanks. For example, the REXX statements:

```
fileid='PROFILE EXEC A1'
parse value (fileid) with file_name . file_type
```

would become in PL/I:

```
fileid = 'PROFILE EXEC A1';
parse value (fileid) with (file_name, . , file_type);
```

All of the expected REXX PARSE statement options are supported including:

1) Comments (balanced /*...*/ pairs like in PL/I) inside the WITH template
2) Literal patterns, e.g., '('
3) Placeholders, that is, the period '.'
4) Positional patterns, e.g., 11, +5, -10
5) Variable patterns, e.g:, (var_name)
6) The UPPER keyword to uppercase the source string before parsing begins

A typical use for this macro is in parsing command arguments and options when writing VM command processors in PL/I. A VM program to plot an HP-GL plot file on a pen plotter might be invoked by an end-user as

```
  PLTHPGL fn <ft <fm>> <( WIDTH www  HEIGHT hhh>,
```

where fn is the (required) file name of the plot file, ft is the (optional) file type (default is HPGL) and fm is the (optional) file mode (default A). WIDTH and DEPTH are options indicating the desired plot size; if not specified, they default to WIDTH 8.5 and HEIGHT 11.0. The beginning of this PL/I application might look something like this:

```
PLTHPGL:
     Proc(Arguments) Options(Main);
        Dcl Arguments Char(100) Varying;
        Dcl Fn Char(8) Initial(' '); /* File name */
        Dcl Ft Char(8) Initial(' '); /* File type */
        Dcl Fm Char(2) Initial(' '); /* File mode */
        Dcl Options Char(100) Varying;
        Dcl Width  Char(4) Initial(' '); /* Plot width */
        Dcl Height Char(4) Initial(' '); /* Plot height */
    /* Break apart the user-specified arguments */
/* get the file id first, substituting defaults */
/* where necessary. Uppercase everything first. */
PARSE UPPER VALUE(Arguments) With(Fn, Ft, Fm, '(', Options);
If Fn = '' Then
   Put List('ERROR, File name must be given');
If Ft = '' Then /* Use default */
   Ft = 'HPGL';
If Fm = '' Then /* Use default */
   Fm = 'A1';
/* Now see what options were given, if any */
Width = ' 8.5';
Height = '11.0';
If Options ¬= '' Then
   Parse UPPER VALUE(Options) With('WIDTH', width, . , 1,
        'HEIGHT', height, . );
```

Using the PARSE macro makes this code easy to write and document for others to understand at a later date.

The fact that a programming construct as sophisticated as the REXX PARSE instruction can be implemented in only a few hundred lines of PL/I preprocessor code clearly demonstrates how powerful that tool really is. It is doubtful that any of the other commonly used preprocessors for third generation languages (e.g., the C preprocessor) could be used as a vehicle to build such a complex yet compact macro. Eberhard used the standard preprocessor facilities of the OS PL/I Version 2 Release 3 compiler; the "new and improved" preprocessor found in the OS/2 PL/I products offers even more usability and functionality. Take a few minutes to study the PL/I macro concepts and facilities and increase your effectiveness when using either the host System/390 PL/I products or the newer OS/2 ones. PL/I -- Productivity Language / One.

Questions and Answers

Question	Answer

I'm trying to declare a structure of size unknown, based on a pointer, L_PTR, passed to me. It points to the following:
 	
amount   fixed bin(15)
element1   char(8)
element2   char(16)
:   :
and so on, where amount tells me how many (element1,element2) pairs there are.
	The REFER option should work:
	
	DCL 1 struct based(L_PTR),
	    3 amount fixed bin(15),
	    3 array (max_amt REFER amount),
	      5 element1 char(8),
	      5 element2 char(16);
	DCL max_amt fixed bin(15) init(?);
	  /* Init with your maximum value */
	
	You can then have:
	 Do i = 1 to amount;
	   do something with element1, element2
	 End;

Question	Answer
I'm creating a window with Visual PL/I for OS/2 that has several entry fields in it. How do I set it up so that I can Tab from one entry field to another?	This takes a few steps! 1. For each entryfield, click Mouse button 2 and select "Styles" from the popup menu. In the "Window Styles" dialog box, select WS_TABSTOP from the "WS_ Styles" listbox. 2. Pick the order in which you want to tab between entry fields. Hold down the shift key and click Mouse button 1 in each entry field, in the <u>reverse</u> order that you want tabbing to occur in. Release the shift key! Go back to the first entry field you selected and click Mouse button 2. Select "Layout" and then "Order group". 3. Set up a link for the window by clicking Mouse button 2 in the client area and selecting "Links". Then, select the WM_CHAR event and the "Tabbing Functions" from the "Library" listbox. Add the "Do group tabbing" code block. That should be all you need!
Can PL/I 2.3 sub-tasks take advantage of VSAM's multiple string reads by using the STRNO sub-parameter in the VSAM AMP parameter?	No. PL/I routines do not check the VSAM control block for this information. Instead, they check their own control blocks, which only allow a single read. Read access would have to be serialized by using either the PL/I WAIT statement or an assembler routine.
I followed the tutorial in PL/I for OS/2 Toolkit Reference V1R1 and created MyProj. When I ran it, I noticed MyProj did not appear in the TaskList. If I minimize MyProj, I have no way of getting it back!	You can get your application into the TaskList by using the FCF_TASKLIST style for your window. This frame creation flag tells Presentation Manager to put your application name in the TaskList. To specify this flag, single click mouse button 2 in the client area for your main window. Select "Styles" from the pop-up menu. A "Window Styles" dialog box should appear. A listbox named "Item Styles" is in the dialog, and from here you can choose FCF_TASKLIST. This will cause the Title bar text for your main window to appear in the TaskList.

I know they're around here somewhere...

Have you ever wondered if there was a place where folks hang out and talk about PL/I? Well, there are quite a few places on-line where PL/I issues are discussed. Here's how to get to some of them:

Compuserve
Enter: *go os2dfl*
Then go to subsection 6: "Rexx and other languages".
This bulletin board is used to discuss OS/2 languages, PL/I among them.

Internet
There is a LISTSERV list that is devoted to discussion of the PL/ I language. To subscribe to it, send a note to:
LISTSERV@UIUCVMD.BITNET (or LISTSERV@VMD.CSO.UIUC.EDU)
containing the line:
SUB PL1-L Your Name

IBM Customer Forum
There is an IBM customer forum devoted to OS/2 PL/I on IBMLink. Even though it is officially dedicated to the OS/2 product, we are in the process of trying to open it up for discussion of all IBM PL/I products. So, if you have a host question, please feel free to ask it on this forum. To access it, do one of the following:
From the IBMLink Main Menu, select option 5, Martlink
From the Martlink menu, select option 2, New Talklink
From the NewTalklink menu, select option 6, OS2bbs1
From the OS2bbs1 menu, select option 1, QAndabbs
From the QAndabbs menu, select option 10, TLpgcat
From the TLpgcat menu, select option 20, OS2PLI
OR
Enter the fastpath *OS2;OS2PLI* from the IBMLink main menu!

IBM PL/I developers monitor all of these. If you know of any other bulletin boards, we'd love to hear about them!

What Next?

If you'd like to continue receiving "The PL/I Connection" newsletter, please let us know. If we don't hear from you, we'll assume you're not interested. You may either mail or Fax your response to one of the following:

PL/I Newsletter
IBM - Santa Teresa Lab.
555 Bailey Avenue, J84/D345
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Fax: (408) 463-4820

The PL/I Connection
Response Form

Yes, I would like to continue receiving the PL/I Newsletter.

Name: ______________________________

Company: ______________________________

Address: ______________________________

____

____

____

____

Phone Number: ______________________________

What PL/I Products do you use? ______________________________

____

Comments: ______________________________

____

____

____

____

____

____

____

____

____

____