The PL/I Connection

Issue 2 IBM Santa Teresa Laboratory March, 1995

CSD's - What and Where are they?

If you're using PL/I for OS/2, here's some information on how to access maintenance. It is available as Corrective Service Diskettes (CSD's) and can be found on-line in several places. There are separate CSD's for the Professional Edition and the Personal Edition. Each CSD incorporates all previous CSD's, and can be applied only to the specified version of the product. For example, it cannot be applied to a demo version. Once you've created diskettes of the CSD's, they can be installed using the A:\SERVICE command. The README files list the fixes that are in the CSD.

CSD #3 is now available. If you intend to use PL/I for OS/2 with **OS/2 Warp**, you must have CSD #3 installed. You can find it in the following locations:

Anonymous FTP

The CSD's are available on 129.34.139.5 or software.watson.ibm.com. They are in the /pub/os2/os2fixes directory. Use the loaddskf utility to make diskettes. The loaddskf utility is also in /pub/os2/os2fixes. The files are named:

- PLPCS3 - Professional Edition
- PLWCS3 - Personal Edition

IBM TalkLink

The CSD's are on TalkLink on the OS/2 Bulletin Board System (OS2BBS) in the Software Library.

CompuServe

The files are in library 6 and are named:

- PLIP31.ZIP through PLIP34.ZIP - Professional Edition, Diskettes 1 to 4
- PLIW31.ZIP through PLIW34.ZIP - Personal Edition, Diskettes 1 to 4

To create the diskettes, unzip each file onto an empty, formatted 1.4MB diskette. Use the OS/2 LABEL command to label each diskette IBMPLICSD1 to IBMPLICSD4.

Other BBS's

The owners of many other OS/2 Bulletin Board Systems around the world are notified of the availability of the PL/I for OS/2 CSD's. They decide whether or not to put them on their system. Look around and see if you can find them. If not, it may be worth asking the BBS owner to make them available. If you do find the CSD's on your system, please send a note to teampli@vnet.ibm.com so we can keep an up-to-date list. We have been told that the latest CSD is available on the EMEA DAP BBS in the LANGUAGE.FIXES file area.

OS2CSD (for IBM internal customers)

Use the TOOLCAT OS2CSD command and retrieve the PLIPCSD3 (Professional Edition) or PLIWCSD3 (Personal Edition) package.

Late Breaking News!! CSD #1 for the PL/I for OS/2 Toolkit is now available as well!!

New PL/I for VSE Redbook

A new IBM Redbook, titled "31-bit Addressing in PL/I for VSE - Getting Started", is now available. The abstract follows:

"This document is unique in its detailed coverage of 31-bit addressing considerations in a system with LE for VSE and PL/I for VSE installed. It provides detailed guidance on setting up and running a system to provide effective initial exploitation of 31-bit addressing for the PL/I user in a VSE/ESA environment. This document was written for systems programmers and administrators of installations that plan to migrate their applications from DOS PL/I to PL/I for VSE in order to gain the advantages of 31-bit exploitation both in batch and on-line. Some knowledge of CICS/VSE, the PL/I language, and 31-bit principles is assumed."

Redbooks are general use technical documents produced by IBM's International Technical Support Organization (ITSO). They are written by experienced IBM professionals from around the world. A list of all recent ITSO Redbooks with abstracts of the books is available on the Internet:

http://www.redbooks.ibm.com/redbooks/itsopub.txt

A catalog of all available Redbooks, sorted by product area is available on:

http://www.redbooks.ibm.com/redbooks/itsopub.cat

These files can also be retrieved by anonymous FTP to *almaden.ibm.com*. They are in the REDBOOKS directory.

If you would like to order a Redbook in the USA., call 1-800-879-2755 or send a FAX to 1-800-284-4721. To order Redbooks from other countries, you can either use the PUBORDER application on HONE or send a note to dkibmbsh@ibmmail.com or to the internal IBM address, BOOKSHOP at DKIBMVM1. The PL/I for VSE book's publication number is:
GG24-4271-00.

If you have any questions about Redbooks, you can send a note to redbook@vnet.ibm.com.

PLIFORM -- The PL/I Source Code Formatter

by Paul Magnussen, Magicon Inc.

Do you have a mass of unreadable legacy code? Are you fed up with maintaining indentation by hand? Do you wish the code in your installation could be made to follow some standard?

If you answered "Yes" to any of the preceding questions, you need PLIFORM, the PL/I reformatting utility from Magicon Inc. PLIFORM takes a PL/I source program as input, and outputs a new version, reformatted to standards you specify.

The need for proper formatting

Programmers realize that proper indentation (reflecting the logical structure of the program) makes a crucial difference to readability. However, establishing and maintaining this indentation is not a trivial task -- many old programs don't observe it at all, or do so in a haphazard fashion. Reformatting code by hand is not only time-consuming and tedious, it can lead to typos which introduce new errors.

What PLIFORM can do

- PLIFORM can make old, unreadable code readable, by formatting it better.
- It can simplify maintenance: even when a program is initially well structured, changes in indentation become necessary as revisions are made.
- It can ensure all code in an installation is formatted to the same standard. Formatting consistency reduces the amount of time required to understand new code.

PLIFORM lets you tune the characteristics of your source to a very fine level of detail. If you use the supplied default parameters, you can reformat your first program in just a few minutes. If you want to fine tune the output, half an hour with the manual should be more than enough to find all of the features you want. Memorizing all the features would take longer -- there are a lot! You can format just one part of a program, or you can format different parts in different ways. You can also skip sections.

PLIFORM was written by Paul Magnussen, a 25-year PL/I veteran and former member of the IBM PL/I compiler team. The first version was produced in 1974. It was rewritten completely in 1977, and since then features have been gradually added and tested (new features are carefully made optional, so in case of any bugs they can easily be turned off).

PLIFORM can turn this:

```
                         DO I = 1 TO N;
              IF OUT_STR = ''
            THEN RETURN ('');
OUT_STR = SUBSTR(OUT_STR,VERIFY(OUT_STR,' '));
         /* Remove leading blanks.   */
           J = INDEX(OUT_STR,' '); IF J > 0
THEN IF I = N THEN RETURN(SUBSTR(OUT_STR,1,J - 1));
  ELSE OUT_STR=SUBSTR(OUT_STR,J);
                       ELSE IF I = N
             THEN RETURN (OUT_STR); ELSE RETURN ('');
END;
```

Into this:

```
DO I = 1 TO N;
  IF OUT_STR = ''
  THEN RETURN ('');
  OUT_STR = SUBSTR(OUT_STR,VERIFY(OUT_STR,' '));
                    /* Remove leading blanks.   */
  J = INDEX(OUT_STR,' ');
  IF J > 0
  THEN
    IF I = N
    THEN RETURN (SUBSTR(OUT_STR,1,J - 1));
    ELSE OUT_STR=SUBSTR(OUT_STR,J);
  ELSE
    IF I = N
    THEN RETURN (OUT_STR);
    ELSE RETURN ('');
END;
```

Availability

PLIFORM is now available for OS/2. Mainframe versions can be supplied by arrangement. For more information, contact Magicon at:

Magicon Inc.
909 University Avenue, Suite 22
Los Gatos
CA 95030-2345
U.S.A.

Tel. (408) 354-7361
Internet: MagiconInc@aol.com
CompuServe: 71601,1404

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 series of articles will address the issues involved in deciding whether to recode PL/I applications to C. It will be presented in four parts. This first part will address the viability of rehosting PL/I applications to open systems as well as the re-engineering option. The next three parts will cover the general concerns of recoding from PL/I to C, the specific concerns, and the costs involved in recompiling vs. recoding.

Moving to Open Systems

Moving legacy applications to open systems is of vital concern to many Information Systems organizations today. Recent surveys of IBM mainframe sites by The Gartner Group and International Data Corporation (IDC) indicate that approximately 70% of these IBM mainframe sites will be moving applications to open systems over the next few years.

There are three alternatives for moving an application from proprietary to open systems:

1. Recoding, which is to convert the existing logic from one language to another. This article will present the issues involved in recoding PL/I applications to C.
2. Re-engineering, which is to redesign and rewrite the application in order to take full advantage of the new environment.
3. Recompiling, which allows the use of the legacy application with minimal modification on the new platform.

Re-engineering is ultimately the right option for certain applications. However, it is a long, expensive, and complicated endeavor. It is the premise of this series of articles that even for systems that need to be re-engineered, recompiling is the right decision for the short term because it gives organizations the ability to gradually re-engineer their applications with less risk. Recoding or recompiling are often looked to as quicker, less risky, and more cost-effective solutions for those systems which will require little or no re-engineering. This series of articles describes the issues relating to recoding versus recompiling.

Staying With PL/I

If your legacy application is written in PL/I, there are many benefits to continuing to use PL/I. PL/I is a language designed by IBM to include the best features of FORTRAN and COBOL in a modern "structured programming" language. Conrad Weisert, of Information Disciplines Inc., said the following in a recent issue of *ACM SIGPLAN Notices*:

"Imagine a general-purpose programming language that offers:

- *better exception handling than Ada*
- *better string handling than Basic*
- *better Input-Output than COBOL*
- *better computation than FORTRAN*
- *better structured flow control than Pascal*
- *better macros than Assembler*
- *better memory management than C*

where "better" means some combination of easier to use, easier to learn, more complete, more reliable, and more fully integrated with the rest of the language. Wouldn't that language be worth looking into as a candidate vehicle for your next big software development effort? Well, that language exists, and it's not some vendor's proprietary "4th generation" wonder, but an established language supported by ANSI and international standards. It's PL/I."

PL/I is a very powerful and a very well designed language. PL/I supports a wide spectrum of data types and storage classes, has extensive support for arrays and record structures, a full arsenal of logic control mechanisms, extensive exception handling, and multiple modes of I/O. The language prefers regularity of constructs and avoids quirky tricks. These are some of the reasons PL/I is often preferable over other languages.

The option of moving PL/I applications to open systems by recompiling is now a viable one due to the availability of PL/I development tools on many open system platforms. Both Liant Software Corporation and IBM Corporation support good PL/I tools on various open system platforms.

Re-engineering Legacy PL/I Applications

Re-engineering, which is to redesign and rewrite an application in order to take full advantage of the new environment, is one of the alternatives to consider in moving a legacy application to open systems.

If your existing application can no longer be enhanced to meet the needs of the users or to take advantage of newer technologies, you most likely will eventually want to redesign it. An advantage of recompiling as your first transition step is that it gives you the time to carefully plan and design the bigger re-engineering effort. You save time on the initial transition of your existing logic and gain time for the re-engineering work.

However, when rewriting your application, you should not too hastily discard your existing PL/I code. If the application is reasonably structured, as many PL/I applications are, it is likely that much of your underlying data processing code can easily be adapted into the new scheme. One big advantage of keeping some of this existing, proven code is that many subtle rules of your business operation are hidden in that code. Things have worked in a certain way for many years, and people have come to depend on that. Problems could arise if this code and these rules are changed.

The other option available for migrating legacy PL/I applications to open systems environments is recoding the PL/I applications to another language. This option will be addressed in future editions of *The PL/I Connection*. The next edition will discuss the general concerns of recoding an application from PL/I to C.

Next Issue....Recoding From PL/I to C - The General Concerns
And Beyond...Recoding From PL/I to C - Specific Concerns
Recompiling vs. Recoding - The Bottom Line

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 openpli@lpi.liant.com.

PL/I in the News!

- PL/I for OS/2 was recently reviewed in "OS/2 Magazine". The review is in the April, 1995 issue starting on page 24.
- An article discussing PL/I and VM is in the latest, March 1995, "Enterprise Systems Journal". It is titled "Swift Energy Co. Refines VM Operating System".

Host PL/I migration aid

by Bob Rasch, IBM PL/I Development
totb@stlvm20.vnet.ibm.com

Background

OS PL/I Version 1 has been withdrawn from Marketing, and it is scheduled to be withdrawn from Service at the end of 1995. Yet there are thousands of customers who have licenses for OS PL/I Version 1. Due to the high level of source code compatibility, we expect that most applications can be migrated to PL/I for MVS & VM or OS PL/I Version 2 without requiring any source code changes. But what can we do to help those customers who have programs that require source code changes?

The answer is that we can provide a migration aid to automate the most prevalent source code changes - the unsupported 48-character set, also called CHARSET(48). Support for CHARSET(48) was dropped in OS PL/I Version 2. Since that time, it has become apparent that more customers have 48-character set source code than we previously believed. Therefore, soon after OS PL/I Version 2 was shipped to the field, I began work on a program to convert unsupported CHARSET(48) source code to supported CHARSET(60) source code.

The first version of this migration aid, called CS48BCD, was shipped as a sample program with OS PL/I Version 2. It is also available to customers through their account representatives, who can receive the package over the IBM network by sending a note to TOTB at STLVM20.

This first version has some limitations that make it less useful that it might be. It is a simple scanner that translates CHARSET(48) keywords into CHARSET(60) keywords. It is unable to recognize embedded preprocessor statements for products like CICS, DB2, and IMS, so it translates anything in these statements that looks like a CHARSET(48) keyword. Unfortunately, the PL/I 48-character set includes keywords like AND, OR, and NOT, which are also used by non-PL/I preprocessors. The first version of the migration aid translates these keywords into PL/I CHARSET(60) symbols, which are not recognized by the non-PL/I preprocessors.

Enhancements

To fix the problem of CICS, DB2 and IMS preprocessor statements, I made extensive modifications to my migration aid. It now does rudimentary lexical analysis, so it has a limited amount of information about the context in which a keyword occurs. Therefore, it will not change words that appear to be CHARSET(48) keywords if they occur in EXEC statements.

Some OS PL/I Version 1 application programs have character string constants that contain the hexadecimal values that are now reserved for graphic shift codes. The code points are X'0E' for shift-out, and X'0F' for shift-in. To migrate such programs, these character string constants must be changed to mixed string constants. I have added support to change character string constants containing these code points to mixed string constants.

Some application programs use PL/I preprocessor statements to generate PL/I source code. If the program uses CHARSET(48), it may have PL/I source code inside preprocessor string constants that must be translated to migrate the application. This is rather difficult to do, because it requires processing the contents of string constants as if they were PL/I source code. I have added support for this feature, to the extent that is feasible. It should only be used for source code that requires it, and it may not work for all cases.

Availability

The new version of the PL/I migration aid is currently in Beta Test. When it is ready for distribution, we plan to make it available as a PTF. There will be no charge for the PL/I migration aid for any customer who has a license for OS PL/I. Further information about the migration aid will be published in this newsletter. You can also call the PL/I Hotline for more information. (The hotline number is given later in the newsletter).

Peter's Performance Tips

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

One of the new features introduced in PL/I for OS/2 was the attribute UNSIGNED. While this was mainly introduced in order to be able to provide PL/I equivalents for C constructs such as unsigned short, the UNSIGNED attribute has some performance implications as well.

One place where the UNSIGNED attribute can make a difference is in DO loops with a variable BY clause. Consider a loop of the following form:

```
dcl (i,j,k,m)   fixed bin;
     do i = j to k by m;
      /* ..... */
     end;
```

Since m, the variable named in the BY clause, is a SIGNED variable, this loop could be running up or down, and the compiler generates extra code accordingly. However, if you know that m is always positive, you could add the UNSIGNED attribute to the declaration for m, and the compiler would be able to generate better and simpler code.

So, this is a continuation on the theme from the last newsletter: the more you tell the compiler, the better your code will perform. But we're not done with this theme yet.

PL/I for OS/2 also introduced the REM built-in function to PL/I. The REM built-in function is the same as the remainder function or operation in many languages and is very similar to the PL/I MOD built-in function. In fact, the MOD function can be defined in terms of the REM function:

```
if x >= 0 then
  mod( x, y ) = rem( x, y );
else
  mod( x, y ) = abs(y) + rem( x, y );
```

The compiler generates code matching the definition above except when the first argument to the MOD built-in has the UNSIGNED attribute, in which case the compiler will generate the simple code for the REM function. So, again, you get better code by telling the compiler all you know.

Finally, as the following example will show, not only can you get better code by telling the compiler all you know, you can also get more readable code.

A positive FIXED BIN value can be divided by a power of 2 by performing a shift. It can then be tempting to write "isrl(x,3)" or "lower2(x,3)" rather than "x/8" when you know that x is nonnegative. However, it would be better to declare x as UNSIGNED, and the compiler will convert "x/8" into a shift for you (even at no optimization). This is better because "x/8" is more readily understood when you or someone else looks at this code 6 months later. Similarly, if you need to divide an expression such as "x+y" (that you know is unsigned) by a power of 2, such as 16, it would be better to write that division as "unsigned(x+y)/16" rather than as "isrl(x+y,4)".

I know they're around here somewhere...

(If you feel like you've seen this before, you have! We're going to republish it so that everyone has a chance to get this information).

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 PL/I on IBMLink. It is in the OS2BBS1, S390, and AIX stores. Within S390, PLI is in the VM and MVS environments. Once you are in one of these environments, you just need to key PLI on the command line, and you will immediately be fastpathed directly into the forum.

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

Travel Reports

DB/2 Expo in NYC

by Fernando Quinones, IBM PL/I Development

The Visual PL/I for OS/2 Toolkit was demoed at Jarvitts Convention Center in beautiful mid-town New York City. The convention was held from December 6, 1994 through December 8, 1994, and catered to DB2 and associated development tools. Visual PL/I falls into this category. The response to the Toolkit was positive. Many attendees that stopped by were impressed by how easy it is to use. Some who have used the Toolkit in the past said that they were addicted and won't go back to doing Presentation Manager development the old way.

If You Liked PLITEST, You'll Love CODE/370

by Bob Davis, IBM PL/I Development
davis@stlvm20.vnet.ibm.com

IBM will soon begin shipping Language Environment Release 4 and has recently shipped CODE/370 Release 2. This combination allows developers to build and maintain PL/I, COBOL, and C applications from a desktop workstation or the host. In effect CODE/370 can help you offload some of your host development from the mainframe to the workstation.

CODE/370 provides integrated tools for developing, porting, and maintaining application programs. It consists of a host-based debugger and optional workstation-based tools. The workstation-based tools operate in an OS/2 windowed environment, in cooperation with a System/370 or System/390 host. This provides the graphical user interface of the workstation along with the power and integrity of the host. CODE/370 gives you all of this and the ability to debug under the CICS/ESA environment while maintaining all the functions of PLITEST.

There are many programmers out there who still have to develop and debug on the host, but would like to take advantage of their workstations. The dual host and workstation offering lets programming shops set their own pace when it comes to adopting the workstation environment for application development of host programs.

CODE/370 consists of two components. One component is a workstation-based programmable editor that offers language sensitive features for PL/I, COBOL, C, JCL and REXX source. The other component is the Debug Tool, which allows the programmer to debug a PL/I, COBOL or C application as it is running in the host environment. The Debug Tool user interface is available on both the workstation and the host.

One feature enables you to monitor the number of times each program statement is executed. "It shows the hot spots in your program." said Bob Rasch (PL/I Development).

CODE/370 helps protect investments and enables growth by providing a common tool for editing, compiling, and debugging new and existing applications written in PL/I, COBOL/370, and C/370. These languages use the services of Language Environment, a run-time common library that provides a consistent run-time environment. CODE/370 also supports the debugging of existing OS PL/I and VS COBOL II applications. This means that CODE/370 can help you migrate applications written in OS PL/I and VS COBOL II to the Language Environment.

PL/I support of CODE/370 release 2 is also provided for LE/370 release 3 through a PTF. For additional information, contact your marketing representative or give us a call on the PL/I Hotline. (The number is given later in the newsletter).

And it really does compile!

Courtesy of Bob Peyser, Piscataway NJ

```
declare aspirin buffered,
        bankruptcy,
        cartoon character,
        dewey decimal,
        five_speed automatic,
        get_your_nose fixed,
        hardly_any static,
        herculean task,
        inferiority complex,
        let_me_give_you_some pointer,
        get_your_wheels aligned,
        nail file,
        naughty bit,
        news print,
        not_to_be_taken internal,
        photo graphic,
        poorly controlled,
        restricted area,
        root_beer float,
        skin condition,
        turkey based,
        union label,
        winery binary;
```

PL/I Hotline

If you'd like more information on IBM's PL/I products, one source of information is the Santa Teresa Laboratory Hotline:

1-800-IBM-4STL (1-800-426-4785)
(Please note that this is USA only).

VisualGen and PL/I

by Mitch Johnson, IBM VisualGen Development
mitchj@vnet.ibm.com
Henry Jicha III, IBM Software Marketing
hjicha_III@vnet.ibm.com

VisualGen is at the heart of IBM's enterprise client/server application development strategy. It is an OS/2 based application development solution for building robust, mission critical client/server and standalone applications that run in a wide variety of workstation and host/server environments. VisualGen includes a visual construction capability for developing true event-driven GUI components and a powerful 4GL scripting language for server or standalone application definition. It also features a complete interactive test facility with logic trace and data probe capability for all application parts including client and server together, or standalone.

VisualGen insulates the developer from environmental differences, and provides middleware to shield development from complex client/server communications protocols. Generated server and standalone applications execute as compiled, optimized 3GL source code that uses runtime libraries like PL/I. GUI client applications have no runtime license cost.

A VisualGen application can call PL/I programs while the application is being developed and tested and after it has been generated. The call format used by the VisualGen application is determined by the contents of a linkage table entry. This entry specifies which linkage type (Operating System, CICS, etc.) is used and how the parameters are passed (by address or in a shared data area).

The following is an example of a linkage table entry which allows a VisualGen application to call a PL/I DLL (VGPLI) using standard OS linkage and parameter passing.

```
:calllink applname=VGPLI bitmode=32
          dllname=VGPLI linktype=dynamic
          parmform=oslink.
```

The source for VGPLI is below:

```
/* PL/I procedure called from VisualGen */
  vgpli: Proc (ws, int01)
         returns(bin fixed(31,0))
         Options( ReEntrant FromAlien
           Linkage(system) NoDescriptor );

   dcl int01  bin fixed (31,0);
   dcl ret    bin fixed (31,0);
   dcl 1 ws,
          2 text.(5)   char(10),
          2 Rc      char(3);

   /* program logic   */

   ret = 0;
   return(ret);
  End vgpli;
```

For more information please see "Developing VisualGen Client/Server Applications", SH23-6563 and "Designing and Developing VisualGen Applications", SH23-6561.

For information on the VisualGen product, call 1-800-426-2279 in the USA, or 919-254-4760 worldwide, (Fax 919-254-4820).

Questions and Answers

Question	Answer
Does PL/I for OS/2 allow the use of dynamic SQL? I need to issue a statement like: EXEC SQL PREPARE stmt_lbl FROM :variable	Yes, you can do this. Anything described in the DB2/2 Programming Guide for the "other" languages can be done with PL/I. This includes things like static SQL with host variables and dynamic SQL with parameter markers.

What is an easy way to compare the contents of two structures and find out if they are equal? One is the original, and the other is declared LIKE the original. I can't use CHAR because they are MIXED structures, and I don't want to have to declare anything with fixed length.	PL/I doesn't allow structures to be compared directly. If you're using PL/I for OS/2 you can do either of the following:
	if unspec(strct1) = unspec(strct2) then...
	OR
	if compare(addr(strct1),addr(strct2),
	stg(strct1)) = 0 then...
	If you're not using PL/I for OS/2, try the following:
	dcl chr1 char(stg(strct1)) based(addr(strct1));
	dcl chr2 char(stg(strct2)) based(addr(strct2));
	if chr1 = chr2 then...

Can a PL/I language program be called from PL/I in a CICS and VSE environment?	External calls to other PL/I subroutines or assembler routines are permitted, as well as standard CICS facilities: EXEC CICS LINK PROGRAM(..) PL/I FETCH is not permitted in the VSE environment.

My PL/I code is being passed a pointer. I need to access storage which precedes that pointer by a certain number of bytes. How can I do that?	If you are using a later version of the compiler (2.3 or OS/2), you can do pointer arithmetic.. If you just need to subtract from a pointer you can simply use:
	p = p - 12;
	If you want to use a pointer expression as a qualifier, you must use the POINTERADD built in:
	pointeradd(p,-12)->b ...
	
	If you are not using a version of the compiler that supports this, you can put the pointer into FIXED BIN(31) format.
	dcl p ptr;
	dcl p2 ptr init(null);
	dcl wk_bin fixed bin(31) based(addr(p2));
	dcl wk_area based(p2);
	p2 = p;
	wk_bin = wk_bin - 12;
	/* perform operations in wk_area */
	
	Note that this approach could cause migration problems if you decide to move to OS/2 PL/I and use the DEFAULT(NONNATIVE) compiler option.

Question	Answer

How can I get my PL/I for OS/2 code to communicate with a COM port?	Here is some sample code showing communication with a modem using COM1. If you have any questions about this, contact Paige Vinall,
	vinall@vnet.ibm.com
	
	`sample: proc options(main);`
	
	`%include os2pli;`
	`%incl_dos='Y';`
	`%include os2;`
	
	`dcl length      builtin;`
	`dcl out_handle HFILE;`
	`dcl data       char(80) varying;`
	`dcl prefix     fixed bin(15) based (addr(data));`
	`dcl written    ULONG;`
	`dcl read       ULONG;`
	`dcl rc         APIRET;`
	`dcl filename   char(80) varyingz;`
	`dcl action     ULONG;`
	
	`filename = 'COM1';`
	`/* Use the standard DOS API to`
	`   communicate with the COM port */`
	`rc= dosopen(addr(filename),addr(out_handle),`
	`        action, 0, file_normal,`
	`        ior(open_action_open_if_exists),`
	`        ior(open_access_ReadWrite,`
	`        open_Share_DenyReadWrite),`
	`        null());`
	
	`put skip list(rc, sourceline());`
	`/* Send the "AT" attention command`
	`   to modem, followed by CRLF */`
	`data = 'AT' || '0a0d'x;`
	`rc = doswrite(out_handle, addr(data)+2,`
	`     length(data), written );`
	
	`put skip list(rc, sourceline());`
	`put skip data(written);`
	`/* Read back the modem's response,`
	`   should be "OK" */`
	`rc = dosread(out_handle, addr(data)+2,`
	`     stg(data)-2, read );`
	`put skip list(rc, sourceline());`
	`prefix = read;`
	`put skip data(data);`
	`put skip data(read);`
	
	`end sample;`

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. If you're a member of Team PL/I or have already responded, you don't need to do so again. 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
You can also send a note to:
teampli@vnet.ibm.com
or
USIB5RLG at IBMMAIL

Trademarks

- IBM, OS/2, CICS/VSE, DB2, Presentation Manager, Language Environment, System/390, CICS/ESA, AIX are registered trademarks of International Business Machines Corporation.
- VSE/ESA, C/370, CICS, IBMLink, System/370, COBOL/370, DB2/2, VisualGen are trademarks of International Business Machines Corporation.
- Liant is a trademark of the Liant Software Corporation.
- All other products and company names are trademarks and/or registered trademarks of their respective holders.

The PL/I Connection Response Form

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

Name:____

Company:______________________________________

Address:______________________________________

Phone Number:_________________________________

Comments:_____________________________________