# The PL/I Connection

Issue 7 IBM Santa Teresa Laboratory June, 1996

Editor: Shauntel Christensen

## *PL/I for Windows is here*

Have you been looking for a way to develop applications in the Windows environment without having to trade in your PL/I code and skill base? Well, wait no longer...PL/I for Windows is now available. Not only does this new product give you a way to protect your PL/I investment, you also get...

- The power of PL/I on Windows 95/NT
- Debug capabilities
- A language sensitive editor
- Performance tuning
- Mainframe compatibility
- And much, much more!

### *Put the power of PL/I on your desktop*

PL/I for Windows offers powerful I/O performance and still maintains the rich implementation of the PL/I language that you expect from IBM.

- Strongly typed enumerations, typed structures and unions, and user-defined types make it easier to identify potential problems at compile time.
- PL/I's various storage classes -- including automatic, static, controlled, defined and based -- increase the flexibility of the language.
- The PACKAGE statement allows you to group related declarations and procedures that share name scope.
- Support for EBCDIC character data, hexadecimal float data, and byte-reversed integers increases mainframe compatibility.

### *Squash those bugs!*

The Windows debugger offers source-level debugging built around a set of core functions designed to let users quickly and efficiently control execution and analyze data, including these features:

- Display and change storage
- Display and change the processor registers
- Display the call stack
- Add and delete simple and complex breakpoints
- Control the execution of multiple threads
- Access source, disassembly, or mixed views of your code

### *Add a little color to your life*

LPEX is a language-sensitive editor that provides not only the basic cut-and-paste text editing features, but customization features and an application programming interface. You can customize LPEX with editing features and functions specific to PL/I files or many other document types.

### *Keep a close eye on performance*

The Performance Analyzer provides you with a tool to monitor the execution of your programs. It offers execution trace analysis and performance tuning. It is designed to help users tune and understand their programs by monitoring program execution and generating a function-by-function trace of the run.

This trace can subsequently be examined by utility programs that graphically display the execution trace. Not only does the analyzer trace procedures in the EXE file, but it traces the entry points to system calls and application DLLs.

### *Pick your preprocessor*

With PL/I for Windows, you can take advantage of preprocessing power:

- Macro facility
- SQL preprocessing
- Include preprocessing

### *Order now!*

So, get on the phone or the Web. In the U.S., fax us your request at (408) 463-4820 or call 1-800-IBM-2468. Outside the U.S., contact your local IBM representative. Or, you can check us out on the Web at:

`http://www.software.ibm.com/ad/pli/plihome.html`

## You missed it!

If you missed GUIDE during the week of June 10, here's a brief description of some of the sessions presented.

***Session 1: PL/I – The migration to LE/370***

This session presented LE/370 migration issues and some new features, such as a solution to the "year 2000" problem. This session will be presented again this summer at SHARE, so if you missed it in June, check out SHARE at the end of July!

***Session 2: PL/I – What's New***

PL/I is a vital part of the IBM language offerings and much development activity is taking place. This session was an opportunity for the PL/I development team to tell you about recent projects like PL/I for Windows NT and Windows 95.

Watch for details about the November GUIDE in San Antonio or the Europe GUIDE this fall. And, if you can't make it to the GUIDE sessions, check out the PL/I sessions at SHARE, July 28-August 2, 1996.

## SHARE and share alike

New Orleans in the summer...AHHH, what could be better? OK, so when it's too hot to be outside, take in a PL/I session at SHARE. The PL/I sessions will be held in the New Orleans Marriott. Here are abstracts for the sessions you won't want to miss!!

***Session 6100***
***Title: PL/I Project Kickoff and Welcoming***
***Chair: Dave Jones (Velocity Software)***

Come to this session to find out what the PL/I Project has planned for the week, and to meet and talk to other PL/I fans and aficionados. Think PL/I's dead, that IBM and other compiler vendors have stopped developing new versions with new features for new platforms? Then stop by and see what kind of PL/I related sessions are scheduled, (including the one on PL/I for Windows) that IBM will be discussing, plus the ones by UniPrise Systems outlining their current and future platform plans. We'll even be giving away a free (!!) copy of PL/I for Windows to some lucky session attendee. It all promises to be a very exciting week for PL/I users! (Monday, 1:30 p.m.)

***Session 6101***
***Title: PL/I - Now We Do Windows (95 and NT)***
***Speaker: Don Smith (IBM)***

PL/I is a vital part of IBM language offerings and much development activity is taking place. This session will be an opportunity for the PL/I development team to tell you about some of our recent projects like PL/I for Windows. (Monday, 3:00 p.m.)

***Session 6102***
***Title: PL/I, The Migration to LE and the Year 2000***
***Speaker: Don Smith (IBM)***

So, you are thinking of migrating your current PL/I applications to the latest LE/370-enabled PL/I? Many issues surrounding this migration will be discussed, including common problems and their solutions, and some new features like a solution to the "year 2000" problem. (Thursday, 4:30 p.m.)

***Session 6103***
***Title: The OS/2 Visual PL/I GUI Builder***
***Speaker: Barry Balk (IBM)***

Witness the power of Visual PL/I in building Graphical User Interfaces (GUIs) for applications that run DB2/2 and IMS Client Server/2. Accelerate and simplify the building of Presentation Manager applications in your favorite language - PL/I. See how you can 'drag and drop' your way into the future with PL/I's visual programming tool. (Tuesday, 9:30 a.m.)

***Session 6104***
***Title: PL/I Running Under MVS on Your x86 PC***
***Speaker: Davinder Athwal (Micro Focus)***

The Micro Focus MVS Workbench provides an integrated development environment for the development of S/370 applications using OS/2 right on your PC. Your PL/I and/or S/370 assembler applications that use IMS DB/DC, DB2, MVS CICS, VSE CICS, ISPF dialog manager, VSAM, IDCAMS, CLISTs and/or MVS sort can be completely developed and tested.

MVS Workbench is as close as you can get to having MVS running under OS/2 (or Windows NT/95) on your PC. It provides the facilities and tools that

## Table of Contents

include TSO and selected commands, interactive and batch (via IKJEFT01) execution of these commands, CLIST support, DB2 MVS Call Attach Facility (DSNALI) with full database integrity (using XDB or DB2/2), emulation of key MVS control blocks such as PSA, CVT, TCB, TIOT, etc., as well as emulation of PL/I S/370 run-time environment so your S/370 Assembler programs that rely on the PL/I environment can still execute on the PC.

In addition, a suite of tools provided with the MVS workbench allows you to do all of your development on the PC without any reliance on a host S/370. Further, your applications will behave the same way as on a S/370 host with respect to EBCDIC, S/370 style integers, S/370 style pointers, and hexadecimal floating point.

With these tools and the state-of-the-art debuggers for PL/I and S/370 assembler, you can increase your productivity when developing, maintaining, and reengineering your applications. (Wednesday, 9:30 a.m.)

***Session 6105***
***Title: A User's Experience with MVS & PL/I on an X86***
***Speaker: Liz Van Hof (Exxon)***

Liz Van Hof, Project Manager, Exxon Company, USA will discuss the implementation of a PC Workbench for mainframe PL/I and COBOL developers at Exxon. Exxon has recently completed the rollout of this Workbench to 100+ developers. A key component of the Workbench is IBM's PL/I compiler and debugger for OS/2. Other components include the MVS Workbench and Revolve from Micro Focus, XDB Workbench from XDB, and TELON PWS from Computer Associates.

The presentation will focus on the process used at Exxon to ensure a successful implementation with an emphasis on key challenges and lessons learned. (Wednesday, 4:30 p.m.)

***Session 6106***
***Title: CODEFORM: Source Code Reformatting Suite***
***Speaker: Paul Magnussen (Magicon, Inc.)***

According to Eric Vesely ("Cobol", ISBN 0-13-854050-0, P.220): "There is a simple software package that yields 10 to 50 percent maintenance productivity improvement. That simple software package is a reformatter."

Magicon Inc. will demonstrate the company's PLIFORM reformatter for PL/I, and discuss the forthcoming versions for COBOL, REXX, Pascal and C/C++ (OS/2, MVS & CMS platforms). Demonstrations of the beta versions will be given. Suggestions for features will be accepted.

Attendees (or others) who would like a preview, or who would like to submit nonconfidential sample programs for demonstration, may contact the company at MagiconInc@aol.com. (Thursday, 11:00 a.m.)

***Session 6107***
***Title: The PL/I Renaissance***
***Speaker: Rick Paul (UniPrise Systems, Inc.)***

On one hand, various analysts are proclaiming the death of the PL/I programming language, citing reasons such as lack of platforms, limited availability of PL/I programmers, and lack of programming tools. On the other, the PL/I programming community, numbering over a hundred thousand programmers worldwide, is now making its presence felt in "cyberspace" communities such as comp.lang.pl1, vocally defending their chosen language and citing numerous advantages over other languages such as COBOL and C. This presentation takes a look at the state of the PL/I market today, including a survey of PL/I compiler offerings and platforms, tools for PL/I application development, organizations providing PL/I training, and other resources available to organizations using PL/I. Emphasis is placed on providing information useful to organizations facing the decision on whether to abandon PL/I altogether, phase it out over time, or continue using PL/I with an expansion of its use. (Tuesday, 11:00 a.m.)

***Session 6108***
***Title: Improving PL/I Development Productivity...***
***Speaker: Rick Paul (UniPrise Systems, Inc.)***

Until recently, writing programs in PL/I meant writing on and for the mainframe. As PL/I developers broaden their horizons from the mainframe to include new platforms such as UNIX, OS/2, Windows NT, and Windows 95, they gain access to a rich heritage of proven cross-platform development solutions. From program editors to source code revision control systems, and beyond program generation tools to new object-oriented development platforms, PL/I developers can now take advantage of capabilities that previously benefited only C and COBOL programmers. This presentation gives an overview of the tools available, introduces several new capabilities, and discusses the benefits to PL/I programmers. (Wednesday, 3:00 p.m.)

***Session 6109***
***Title: Reengineering PL/I Apps. with Analyzer PL/I***
***Speaker: Charles Stump (Leverage Tech., Inc.)***

According to some estimates, 80% of all programming resources are devoted to maintaining and reengineering existing source code. The high cost of maintenance frequently prevents conversion to new software/hardware technologies.

Reengineering requires analyzing and understanding large bodies of existing source code. This code typically exhibits several of the following characteristics that make it difficult to maintain: written in old languages, running on obsolete platforms, use of old DBMS, no (or poor) documentation, etc.

Large scale reengineering projects are dominated by things such as unique hardware requirements, company-specific reengineering protocols, and mixed-language systems.

Therefore, shrink-wrapped CASE tools alone rarely satisfy users' needs. You need shrink-wrapped reengineering capabilities plus the ability to customize. PL/I Analyzer TM, a static analyzer for PL/I code, in conjunction with Software Refinery TM, a meta-CASE environment tailored to code analysis and transformation, provides a powerful and flexible environment for software maintenance and reengineering. (Tuesday, 4:30 p.m.)

***Session 6110***
***Title: PL/I Requirements Discussion and Voting***
***Chair: Dave Jones (Velocity Software)***

Vendors are always eager to hear what new functions and features their users need, and this is your opportunity to let IBM know what you think is missing or needs to be added to their suite of PL/I tools on *all* platforms where PL/I is supported: OS/2, AIX, MVS, VM, and now, Windows 95/NT. We'll also review the status of previously submitted requirements. Make your voice heard and your opinions count as this robust language moves into the 21st century. (Friday, 8:00 a.m.)

***Session 6111***
***Title: Object-Oriented Programming in PL/I***
***Speaker: Richard Smedley (SABRE Decision Tech.)***
***Speaker: Dale Smith (SABRE Decision Tech.)***

Object-Oriented Programming in PL/I describes the translation from generic object-oriented language constructs to working PL/I code. Pragmatic definitions for "class", "object", "inheritance", and "polymorphism" are presented using programming language constructs instead of abstract discussions. These critical object-oriented concepts are explained using the language programmers understand -- working code. For years, programmers in the workstation world have enjoyed the benefits of object-oriented programming using languages widely available on these platforms. This has not been the case for mainframe developers. Using the mechanisms detailed in Object-Oriented Programming in PL/I, commercial programmers can now enjoy the benefits of object-oriented analysis and design techniques and translate their designs directly to PL/I code. (Thursday, 9:30 a.m.)

For more information about SHARE or the sessions being presented at SHARE, point your Web browser to: `http://www.share.org.`

# *PCR - An interface between PL/I, CMS, and REXX*

*by Dave Jones, djones@starbase.neosoft.com*

*This is the last of three parts. The first part appeared in the December, 1995 issue of "The PL/I Connection" and contained introductory material and the syntax of the PCR command. The second part appeared in the March, 1996 issue.*

## *Part 3 - PL/I parameter list*

PL/I programs compiled with SYSTEM(CMS) will receive the normal CHAR(100) VARYING as the first and only parameter, where programs compiled with SYSTEM(MVS) will receive a pointer to the PCRCB as the first and only parameter. The pointer to PCRCB can also be obtained by calling PCRGETC. The use of PCRCB is required by REXX functions and subroutines, but is otherwise not necessary.

## *Accessing REXX arguments*

When called as a REXX function or subroutine, there are 0 or more arguments pointed to by PCRCB_RX_arglist. The following code shows how to access them. The data can be modified by the PL/I program, if needed, perhaps to translate them to upper case.

```
/* Get access to PCRCB (if not already done) */
Call PCRGETC(PCRCB_ptr);

/* Loop through the parms */
RXARG_ptr = PCRCB_RX_arglist;
do i=1 to PCRCB_RX_argcnt;
  /* To access or test first character */
  if RXARG_c1 = '*' ...

  /*To access up to 256 bytes */
  if SUBSTR(RXARG_c256,1,RXARG_len)...

  /*To access up to 32767 bytes */
  /* (Less efficient, but more complete) */
  if SUBSTR(RXARG_c32767,1,RXARG_len)...

  /*Move to next arg */
  RXARG_ptr=RXARG_ptr+STORAGE(RXARG_struct);
end;
```

## Keeping data around between calls

Any static variables that the PL/I program alters will retain their last value between calls. However, any storage allocated via PL/I allocates that the PL/I program does not free will be freed for you when you return your caller. Thus, to allocate and keep storage between calls, the PL/I program must use alternate ALLOC/FREE routines. PCR provides two entry points, PCRSTOR_OBTAIN and PCRSTOR_RELEASE to allocate and free CMS memory. The following code shows their use:

```
/* Must modify the entry address for */
/*   the PL/I entry variables        */
ENTRYADDR(PCRSTOR_OBTAIN)=PCRCB_STOR_OBTAIN;
ENTRYADDR(PCRSTOR_RELEASE)=PCRCB_STOR_RELEASE;

/* Allocate 100000 bytes of storage */
ptr = PCRSTOR_OBTAIN(100000);

/* Or allocate enough to fill a structure */
ptr=PCRSTOR_OBTAIN(STORAGE(some_based_struct));

/* Save it until next call, method 1 */
some_static_ptr = ptr;

/* Save it until next call, method 2 */
PCRCB_local_ptr = ptr;

/* Allow other PL/I routines to access it */
PCRCB_global_ptr = ptr;
 ...
/* Here's how to free it */
call PCRSTOR_RELEASE(ptr);
```

## Be careful with immediate commands

Defining PL/I as immediate commands can cause problems, depending on what the PL/I program is doing. No I/O should be done. Remember that PL/I might need to load dynamic code from text/load libs, which would mean doing I/O.

## SYSTEM(MVS) verses SYSTEM(CMS)

If your PL/I program is compiled with SYSTEM(MVS), it will share a common PL/I environment with any other PL/I SYSTEM(MVS) programs managed by PCR, thus making execution faster. This common runtime environment support is based on the PL/I preinitialization interface provided in the OS PL/I Version 2 Release 3 product (program number 5668-910).

If your PL/I program is compiled with SYSTEM(CMS), it will create and use a new PL/I environment upon each call.

*The PCR package is available at no cost and can be distributed.*

# Peter's performance tips

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

Now that Version 1 Release 2 of PL/I for OS/2 and PL/I for Windows are available, I'd like to alert you to some differences from the past PL/I releases. This material is also documented in the product manuals and most of it is in the readme file, but I think it is important enough to repeat here. These differences are important and will matter to you. I have tried to list these considerations in an order that follows the steps of compiling, linking, and running.

### All modules must be recompiled

This new release contains a significantly different runtime, which is smaller and faster than the previous runtime. Unfortunately, it requires that all code be recompiled. Do not build applications with some code compiled with one of the previous releases and some with this release.

Also, when you are recompiling your programs, note that DLLs and multithreaded applications require some new compile-time options as described in this article.

### Building DLLs requires the DLLINIT option

This new release requires that every DLL contain at least one program compiled with the new compile-time option DLLINIT. This option inserts a reference to a library routine into your object file, thus forcing the linker to link in this routine which is needed by the operating system to initialize the DLL properly. There is no performance penalty and you could compile every program in a DLL with the DLLINIT option.

If you do not compile at least one program in your DLL with the DLLINIT option, your DLL will not link. You will get a message from the linker saying that your program has 'no starting address'. If you are building an EXE, do not use the DLLINIT option to compile any program linked into the EXE.

### Multithreaded programs require the LIBS(MULTI) option

The previous release had one set of libraries that supported both single-threaded and multi-threaded applications. In order to improve the performance of single-threaded applications, the new release has a set of libraries for single-threaded applications and a separate set for multi-threaded applications. This is reflected in the naming convention of our `.lib` and `.dll` files: if the fifth letter is an 'S', the part is for single-threaded applications; if the fifth letter is an 'M', it's for multi-threaded applications.

By default, the compiler assumes your application is single-threaded and puts references to the corresponding libraries in your object files. If your application is multi-threaded, you want the OBJs to point to the multi-threaded libraries instead. To ensure that they do, compile all your code with the option LIBS(MULTI).

### *Use of DATE is flagged*

Programs that use the DATE built-in function now compile with a return code of 4 because the compiler now generates a warning message indicating that such a program will fail after the year 1999.

The compiler also flags with an informational message any use of the constants 19, 365 or '19'. Their presence indicates that the containing program is probably doing some date calculations, and this message allows you to find these programs so that you determine if they will still work after the year 1999.

The compiler user exit can be used to up the severity of either of these messages or to track which modules generate them. If there are other checks that you would like the compiler to make in order to detect possible year-2000 problems, please let me know.

### *ILINK must be used*

The object format of the object files generated by the compiler is different than that supported by LINK386, and consequently, your OBJs must be linked using ILINK.

By default, ILINK uses a 'free format' that is different than the format used by LINK386. While this format is nicer (and is the only format supported by the Windows version of ILINK), it might take you a little while to get used to it. If you want to use the LINK386 format, invoke ILINK with the /NOFREE option.

By default, ILINK is case sensitive while LINK386 wasn't. If you use the /IGNORECASE option with ILINK, it will no longer be case sensitive. More information on linking is available in the Programming Guide.

### *The SIZE condition occurs more often*

With some of the recent CSDs, the compiler has changed the handling of the INVALIDOP, OVERFLOW, SIZE and ZERODIVIDE conditions. Disabling these conditions means now only that the compiler will not generate special code to detect them. If they occur in hardware code or in library code, the condition will be raised and the appropriate ON-unit driven.

You may not notice this difference except that you might see the SIZE condition raised in places you didn't see it raised before. For example, if you assign a character variable to a fixed dec(5) variable, the SIZE condition would be raised in library code if the character variable were bigger than 99,999. In the past, this was ignored if SIZE was disabled. Now it is not ignored, and the ERROR condition is raised if you don't have an ON SIZE block or if you don't issue a goto out of the one you do have.

## *How you can reach us*

We continue to receive addresses from PL/I enthusiasts interested in receiving *The PL/I Connection* newsletter. Thanks for passing the word along. If you know someone who uses PL/I and might be interested in receiving this newsletter, 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 noticed that IBM has released DB2/2 V2.2 for OS/2. Does PL/I for OS/2 support it? If not, what is the highest level? | PL/I for OS/2 (and for AIX) work with DB2 V2.x as well as with the earlier versions of DB2/2. However, not all the V2.x function is supported. We support DUOW and some V2.x compile/bind options. We do not support BLOBs, compound SQL, or triggers. |
| Is there a way to get a 4-digit year field from DATE (for the year 2000+)? | PL/I V2R3 and later support the DATETIME built-in function which returns a 4-digit year. |
| If I have a main program written in 31 bit addressing mode and I want to dynamically call another program (an assembler program, for example) written in 24 bit addressing mode, is it possible to pass parameters or data to the called program? | The answer is YES, but with the following restrictions:<br><br>• PL/I supports dynamic calls via FETCH. However, you cannot fetch COBOL, FORTRAN, or C. As such, you are limited to fetching an assembler program or another PL/I program.<br><br>• The (main) program may have AMODE 31 but it must have RMODE=24 if it is to FETCH a program with AMODE=24.<br><br>• It is the user's responsibility to ensure that all parameters reside below the line so that they are addressable in AMODE=24. |
| Does anybody have experience calling a REXX exec from PL/I for OS/2? | You can invoke a REXX exec from OS/2 PL/I using the new SYSTEM built-in function. You must compile such a program with LANGLVL(SAA2). The following line would invoke the .exe or .cmd named TRYTHIS with the parm string PARMS:<br><br>`rc = system( 'trythis parms' );`<br><br>`rc` would hold the return code from the .exe or .cmd. |

| | |
|---|---|
| Is LE a language like COBOL, PL/I or C? | Language Environment (program number 5688-198) is not a language. It is the common run-time environment for the following language compilers: PL/I for MVS & VM, C/370, C/C++ for MVS/ESA, and COBOL for MVS & VM. You must have Language Environment installed on your system before you can use any of the above compiler products. Language Environment is especially useful for interlanguage communications. |
| My customer wants to migrate many modules from the host to OS/2. In the host source code, he as an exclamation mark as an operator for the logical "or" and concatenation. Is there a simple and fast method to change them to the symbol used on the workstation. | To change the symbol for the logical "or" operator, you can use the OR compile-time option from the PL/I for OS/2 compiler. For example, OR('\|') is the default PL/I logical "or" operator, code point 7C hexadecimal, and OR('!') changes it to the exclamation point. You can do exactly the same sorts of things with the NOT compile-time option as well. A good place to put these options is in the *PROCESS statement:<br>`*PROCESS ... OR ('!') NOT('¬') ...` |
| Because the COBOL and PL/I conversions are not carried out at the same speed, we seem to have some problems. One question we have is: can we run COBOL/MVS programs without a common run-time environment (LE/370)? This would give us the opportunity to convert COBOL much faster, if we can continue to use separate COBOL/MVS and PL/I 2.3 run-time libraries. | The answer is that programs compiled with COBOL for MVS & VM require the LE run-time library. This is also true for programs compiled with PL/I for MVS & VM. You can do a COBOL-only migration by only installing the CEL and COBOL components of LE, but in any migration scenario you must separate out the ILC applications (mixed COBOL and PL/I) and migrate them separately.<br><br>The easiest way to do this is to migrate your ILC COBOL and PL/I programs to VS COBOL II R4 and your PL/I to OS PL/I 2.3, and link them with the PL/I migration aid. Then you have all your ILC applications in a state that will run on either old or new libraries. |

# *Compiling DB2 and CICS Applications on AIX*

*by Barbara Yu, IBM PL/I Development*

The following brief description is the answer to a request for help compiling DB2 and CICS applications on AIX. There are two ways you can compile a DB2 and CICS application.

## *Use the `pli` command*

1. Add the `pp` compile-time option at the beginning of your program. For example:

```
*PROCESS PP(CICS(NOEDF...) SQL('DBNAME(PLITEST) S PRINT' MACRO);
```

2. Use the `pli` command to compile and link the program xxx.

```
pli -I/usr/lpp/cics/include -qsystem=CICS -o xxx.ibmpli
    -bI:/usr/lpp/cics/lib/cicsprIBMPLI.exp -eplicics
    -L/usr/lib/dce -ldcelibc_r -ldcepthreads -ldb2 -lplishr_r -lr_c xxx.pli
```

## *Use the `cicstcl` command*

1. Use the `pli` command to go through the `db2` preprocessor by putting the following statement in your program:

```
*PROCESS PP(SQL('DBNAME(PLITEST) S PRINT' MACRO) MDECK;
```

This will generate the `xxx.dek` file.

2. Delete the `pp(sql)` and `nopp` compile-time options from the *PROCESS statement in your `xxx.dek` program and rename the program to `xxx.pli`.

3. Add the `-ldb2` reference in the `/usr/lpp/cics/lib/cicsportable.fnc` in front of the `-lplishr_r`.

4. Enter the following:

```
cicstcl -lIBMPLI xxx.pli
```

Remember that you need to have DB2 running in order to use the DB2 preprocessor.