SC26-4121-1
File No. S370-34

Program Product

OS PL/I Optimizing Compiler: Installation Guide for MVS

Optimizing Compiler	5734-PL1
Resident Library	5734-LM4
Transient Library	5734-LM5

(These program products are also available as composite package 5734-PL3)

IBM

SC26-4121-1
File No. S370-34

Program Product

OS PL/I Optimizing Compiler: Installation Guide for MVS

Optimizing Compiler	5734-PL1
Resident Library	5734-LM4
Transient Library	5734-LM5

(These program products are also available as composite package 5734-PL3)

Release 5 (5.0 and 5.1)

IBM

Second Edition (September 1985)

This edition applies to Releases 5.0 and 5.1 of:

- OS PL/I Optimizing Compiler, Program Product 5734-PL1
- OS PL/I Resident Library, Program Product 5734-LM4
- OS PL/I Transient Library, Program Product 5734-LM5

and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

The changes for this edition are summarized under "Summary of Amendments" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Preface

This publication is primarily for readers who are responsible for the installation of the OS PL/I Optimizing Compiler, OS PL/I Resident Library, and OS PL/I Transient Library. The book describes how to install these three products under the IBM MVS and MVS/XA Operating Systems. To install this product under the IBM Virtual Machine/System Product (VM/SP) Conversational Monitor System (CMS), see *OS PL/I Optimizing Compiler: Installation Guide for CMS*, SC26-4122.

You should already be familiar with the MVS or MVS/XA system used at your site, with the publications that describe that system, and with its job control language. You should also be familiar with and know how to use Release 4 and any subsequent releases of the System Modification Program (SMP) or the System Modification Program Extended (SMP/E).

How this Book Is Organized

The OS PL/I transient library, resident library, and optimizing compiler function under the control of the IBM MVS and MVS/XA Operating Systems to compile and execute programs written in PL/I. This book describes how to install the transient and resident libraries and the compiler under these systems using Release 4.0 and all subsequent releases of the System Modification Program (SMP) or the System Modification Program Extended (SMP/E). This publication is divided into the following parts:

- Chapter 1 gives details of system requirements, program product distribution, and a summary of the installation procedure for all three program products.
- Chapter 2 through Chapter 5 give information on the installation procedures.
  - Chapter 2 covers the installation of the transient and resident libraries and the compiler, using the composite distribution tape.
  - Chapter 3 covers installation of the transient library.
  - Chapter 4 covers installation of the resident library.
  - Chapter 5 covers installation of the optimizing compiler.
- Chapter 6 describes how to run the installation verification program (IVP).
- Chapter 7 describes the extra steps necessary to complete the installation of OS PL/I's support for CICS.

- Chapter 8 describes how to change the IBM-supplied execution-time and compiler option defaults according to your site's needs.
- Chapter 9 describes how to tune the compiler and libraries to your site's particular needs.

Related Publications

For more complete and current information about the actual installation process for the OS PL/I Optimizing Compiler and Libraries, you must use the *Program Directory* shipped with the product distribution tape(s). In addition, you might want to consult the publications listed below.

A number of publications are referred to in this manual by generic names, such as "Assembler Language publications." In such cases, the actual manual you require will depend on your operating system.

The following manuals might be useful during the installation of the optimizing compiler and the libraries under an IBM MVS or MVS/XA operating system.

OS PL/I books:

OS PL/I Optimizing Compiler: Programmer's Guide, SC33-0006

OS PL/I Optimizing Compiler: Messages, SC33-0027

OS PL/I Optimizing Compiler: TSO User's Guide, SC33-0029

OS and DOS PL/I Language Reference Manual, GC26-3977

SMP books:

Operating System/Virtual Storage (OS/VS) System Modification Program (SMP) System Programmer's Guide, GC28-0673

OS/VS SMP Messages and Codes, GC38-1047

System Modification Program/Extended (SMP/E) Reference, SC28-1107

SMP/E User's Guide, SC28-1302

SMP/E Terminal User's Guide, SC28-1109

SMP/E Messages and Codes, GC28-1108

CICS books:

Customer Information Control System/Operating System/Virtual Storage (CICS/OS/VS) Version 1 Release 6 Modification 1: Installation and Operations Guide, SC33-0172

CICS/OS/VS Version 1 Release 6: Application Programmer's Reference Manual (Command Level), SC33-0077

CICS/OS/VS Version 1 Release 6: Application Programmer's Reference Manual (Macro Level), SC33-0079

MVS and MVS/XA books:

Operating System/Virtual System 2 (OS/VS2) System Programming Library: System Generation Reference, GC26-3792

OS/VS2 System Generation Reference Supplement for MVS/SP, GD26-6027

OS/VS2 System Programming Library: Initialization and Tuning Guide, GC28-0681

Multiple Virtual Systems/370 (MVS/370) Utilities, GC26-4065

Multiple Virtual Systems/Extended Architecture (MVS/XA) System Generation Reference, GC26-4009

MVS/XA Utilities, GC26-4018

MVS/XA System Programming Library: Initialization and Tuning Guide, GC28-1149

MVS/XA System Programming Library: JES2 Initialization and Tuning Guide, SC23-0065

MVS/XA System Programming Library: JES3 Initialization and Tuning Guide, SC23-0059

MVS/XA Conversion Notebook, GC28-1143

MVS/XA System Programming Library: 31-Bit Addressing, GC28-1158

Summary of Amendments

September 1985

CICS Support

OS PL/I Optimizing Compiler, Release 5.1, provides support for the IBM Customer Information Control System/Operating System/Virtual Storage (CICS/OS/VS), Version 1 Release 6.1, under the IBM MVS/Extended Architecture (MVS/XA) system and MVS/System Product (MVS/SP) Version 1 Release 3.

Specifically, OS PL/I Release 5.1 provides support for PL/I programs using the CICS Command Level Interface in both 24-bit and 31-bit addressing modes and the CICS Macro Level Interface in 24-bit addressing mode, only.

CMS Support

OS PL/I Optimizing Compiler, Release 5.1, supports the Conversational Monitor System (CMS) of the Virtual Machine/System Product (VM/SP) Release 3 and any subsequent releases. The installation process for OS PL/I under CMS is described in *OS PL/I Optimizing Compiler: Installation Guide for CMS.*

Book Reorganization

This manual has been reorganized to improve its usability.

- The chapter describing how to install the composite product tape has been placed before the chapters on installing the individual product component tapes.
- Information on running the sample program to verify your installation has been moved to Chapter 6, "Running the Installation Verification Program" on page 47.
- Information on CICS support has been placed in Chapter 7, "Completing Installation of PL/I Support for CICS" on page 51.

- Discussions of how to change IBM-supplied option defaults have been moved to a separate chapter, Chapter 8, "Changing Option Defaults and Customizing PL/I" on page 55.
  The discussion of how to modify the PL/I error exit routine IBMBEER has also been moved to Chapter 8.

Contents

Figures

Chapter 1. General Information

This chapter contains the following information:

- A list of the minimum system requirements for installing and using the transient library, resident library, and compiler, under the control of an IBM MVS or MVS/Extended Architecture (MVS/XA) Operating System.

- A brief description of the new programming features introduced with Release 5.0 of the OS PL/I Optimizing Compiler.

- A summary of the installation process.

This chapter begins by defining the minimum system requirements for using the transient library, resident library, and optimizing compiler under the control of the IBM MVS or MVS/XA Operating System. It describes the programming features introduced with Release 5.0 of the OS PL/I Optimizing Compiler and the installation-time and execution-time options you can use to implement them.

It lists the major features of the installation process and summarizes the installation steps and the preparatory jobs you need to install the three products.

The compiler and its libraries are distributed by IBM as an SMP installation job on magnetic tape(s). The process of installing the product involves executing SMP to place it in the appropriate program libraries.

Before you can add the compiler and its libraries to an MVS or MVS/XA system, that system must be generated to meet the minimum system requirements. For information on MVS or MVS/XA system generation, see the *System Generation Reference* for your operating system.

Interrelease Compatibility

Different release levels of the optimizing compiler and libraries will produce compatible executions, provided that:

- The release and service levels of the transient library are equal to or greater than those of the resident library.

- The release and service levels of the resident library are equal to or greater than those of the compiler.

CICS—PL/I

OS PL/I Release 5.1 provides support for PL/I command level application programs running under CICS/OS/VS 1.6.1 to use the 31-bit addressing capabilities of the MVS/XA operating system. It enables PL/I programs to run under CICS/OS/VS Version 1 Release 6 Modification 1 using the CICS Command Level Interface in 31-bit mode under MVS/XA and the CICS Command or Macro Level Interface in 24-bit mode under MVS/XA or MVS/SP 1.3.

If you have PL/I program load modules that run using the PL/I interface module (DFHPL1OI) from OS PL/I release 3.1 or 4, and the CICS command level interface module (DFHEPI) from CICS release 1.5, or 1.6, they will continue to run under CICS/OS/VS release 1.6.1, using the OS PL/I release 5.1 transient library CICS support modules contained in module DFHSAP.

System Requirements

This section summarizes the machine and operating system requirements for the transient library, resident library, and compiler.

Machine Requirements

The minimum machine requirements for installation of the optimizing compiler are:

- An IBM System/370, 303x, 308x, or 43xx processor, and
- A magnetic tape drive on which you can run the distribution tape(s)

Auxiliary Storage Requirements

Storage estimates for SMP are given in *OS/VS System Modification Program (SMP) System Programmer's Guide.* Storage estimates for SMP/E are given in *System Modification Program Extended (SMP/E) Reference.*

The storage estimates for the OS PL/I Optimizing Compiler and libraries are provided in the *Program Directory.*

Operating System Requirements

Subject to requirements given below, the compiler and its libraries will operate under the following systems:

- OS/VS2 (MVS) Version 3 Release 8 with MVS/SP Version 1 Release 3,
- MVS/SP Version 2 Release 1, with MVS/XA DFP Version 1 Release 1,

and any subsequent versions, releases, and modifications.

OS PL/I also operates under the IBM Virtual Machine/System Product (VM/SP) Release 3 operating system. If you are installing under VM/SP, use *OS PL/I Optimizing Compiler: Installation Guide for CMS* instead of this book.

Your operating system must include the following:

- An eligible automatic restart user abend code of 4092 if the PL/I Checkpoint/Restart interface is to be used, and automatic restarts are to be forced by PL/I programs during execution using the PLIREST statement.
- IEAXPSIM module must be in the link list if you are running under MVS/SP Version 1 Release 3 and your machine does not support extended floating point.
- An XF-level assembler is needed to process the macro instructions used to change the PL/I compiler and/or execution-time option defaults.
- Assembler H, Version 2 or later, is needed for the transient library running under MVS/XA, to change the PL/I execution-time option defaults.
- CICS/OS/VS Version 1 Release 6 Modification 1 with UP90207 for HCI1612 (MVS/370) or UP90208 for HCI1613 (MVS/XA) or later, if you plan to use OS PL/I Release 5.1's support for CICS.
- IMS/VS Version 1 Release 2 or later if you plan to use the improved IMS-PL/I error handling support.
- IMS/VS Version 1 Release 3 or later if you need support for 31-bit PL/I transactions under IMS/VS.

Program Product Distribution

The machine-readable program materials you need to install the library and compiler program products are distributed in the following ways:

1. The transient library, resident library, and compiler installation material (in that order) are distributed on a single tape as a composite package (orderable as program product number 5734-PL3).
2. The transient library, resident library, and compiler installation material are also distributed on three separate tapes, orderable as program products:
    - Transient Library: 5734-LM5
    - Resident Library: 5734-LM4
    - Optimizing Compiler: 5734-PL1

    You may not install the resident library without the transient library; you may not install the compiler without both transient and resident libraries.

Format of Distributed Tape(s)

The distributed tape(s) will be one of the following:

9-track, 1600 b.p.i.
9-track, 6250 b.p.i.

If you order the composite package, you will receive a multifile installation tape in SMP format with one set of the following files for each product—transient library, resident library, and compiler:

- SMP control statements.
- ++JCLIN file. This is a job stream.
- IEBCOPY unloaded data sets comprising product load modules, cataloged procedures, installation jobs, and macros.

If you order the program products on separate tapes, each product comes in the form of a multifile installation tape in SMP format. The SMP tape has the set of files listed above.

New Programming Features

Beginning with Release 5.0, the OS PL/I optimizing compiler and libraries provide support for utilization of storage above 16 megabytes under MVS/SP Version 2 Release 1, with MVS/XA DFP Version 1 Release 1.

New Installation-Time Option

A new option, 'ARCH', specified at the time of installation, indicates whether the system onto which you are installing the products supports both 31-bit and 24-bit addressing or just 24-bit addressing.

New Execution-Time Options

Three new execution-time options have been added to utilize extended architecture support:

HEAP
enables you to direct PL/I to isolate storage allocated to CONTROLLED and dynamically allocated BASED variables from all other PL/I storage, and to specify size and location of the HEAP area.

ISAINC
enables you to specify a minimum size for increments of the ISA over the initial allocation. Use of this option can help decrease the number of GETMAINs performed during execution of a task.

TASKHEAP
used only in a multitasking environment, enables you to direct PL/I to create separate HEAP areas for all subtasks, with their own size and location parameters.

Information on specifying these options and establishing default values for them is in Chapter 8, "Changing Option Defaults and Customizing PL/I" on page 55.

Installation Process

The process of installing the OS PL/I optimizing compiler and libraries includes some features that are designed to simplify the process and enable you to more easily tune the products according to the needs of your site.

- You can select or eliminate, by specifying the appropriate function modification identifier (FMID), certain optional facilities included on the distribution tape(s). These optional features include TSO support for the compiler, multitasking support during program execution, and, if you are installing release 5.1, CICS support for PL/I command level application programs.

- You can obtain JCL for the various installation steps directly from the distribution tape(s). You can install these jobs into a temporary data set (which we have named PLI.INSTALL) and erase this data set when your installation is complete. When you run the SMP ACCEPT job, the installation JCL is stored with the rest of the distributed product materials in your distribution libraries.

  You can copy and make minor changes to these jobs and use them when you run your SMP installation job. Each individual product tape includes a set of installation jobs designed and named for that product. The composite tape includes all three sets of installation jobs plus a set of jobs designed to let you install all three products at one time.

- You can use either SMP or SMP/E to install the products. Installation JCL is provided for both SMP products.

- IBM-supplied execution-time and compiler option default values are automatically established during SMP RECEIVE and APPLY processing. If these default values are satisfactory for your site, you will not need to take any further action. If you want to change some of the defaults, you will be able to do so after installation is complete by using an SMP USERMOD process.

- If you are using the composite product tape (all three components on one tape), the installation jobs are coded to RECEIVE the three components and then to APPLY them together.

If you are using individual product tapes, you must run your installation jobs up to and including the SMP RECEIVE job for each of the tapes, in the following sequence:

1. The transient library first
2. The resident library next, if your site needs it, and
3. The compiler last, if your site needs it.

After you have RECEIVEd all the tape(s) you have chosen for your site, you *must* APPLY all the components at the same time.

Installation Steps

You can install the transient library, resident library, and compiler at any time after your operating system is initially generated.

First you decide which product(s) your site needs. You can install:

- The transient library, or
- The transient library and the resident library, or
- The transient library, the resident library, and the compiler.

Next, you choose the optional components your site needs. You can choose and install:

- Multitasking support for the transient and resident libraries
- CICS support for the transient and resident libraries, if you are installing release 5.1.
- TSO support for the compiler

Next you can obtain from the distribution tape(s) the installation JCL, which you can modify according to your site's needs. There are installation jobs for both SMP and SMP/E.

Finally, you can execute the modified installation jobs to allocate product and SMP data sets, establish a cataloged procedure to be used during the installation process, install the product(s) using SMP, and change the execution-time and/or compiler option defaults, if necessary.

Installing Individual Product Tapes

If you are using individual product tapes to install one or more of the products, you will use installation jobs designed and named for the individual product. Each tape's installation jobs have names that begin with a prefix representing the product on that tape:

Product Name	Job Name Prefix
Transient Library	L5
Resident Library	L4
Compiler	P1

Note: You should keep the distribution materials you received from IBM as backup in case you use system libraries for PL/I modules and re-creation of your operating system destroys and reallocates the system libraries. Also, if you do not install all the PL/I products at this time (if, for example, you install only the transient library), and later decide to install an additional PL/I product at the same release level (the resident library, for example), you will have to reinstall the previously installed product(s) at the same time that you install the additional product(s).

Installing the Composite Tape

If you are using the composite tape, you will find all three of the above listed sets of installation jobs on the tape. The jobs you will actually use, however, are designed to install all three products at once. These jobs begin with the prefix 'P3', and are located with the compiler installation files on the composite tape.

Chapter 2, "Installing the Composite Product" on page 13, explains how to use the composite tape installation jobs (named 'P3xxxx') to install the transient and resident libraries and the compiler all at one time.

Multiple Program Product Installation

Attempting to RECEIVE the transient library, resident library, and compiler at the same time in a multiprogramming environment may, during installation, cause problems because of data set contention. You can prevent unnecessary enqueuing of jobs on a single data set by scheduling all SMP RECEIVE jobs to a single class and starting only one initiator to that class. This forces sequential processing of program product installation job streams, but allows maximum multiprogramming with your usual jobs. You *must* APPLY all the PL/I products you have selected for your site at the same time and with a single job.

Preparing for Installation

There are several decisions you must make before beginning the installation process.

Choosing Optional Subcomponents

Before installing the transient or resident libraries or the composite product, you must decide whether you want multitasking support and/or CICS support at your site. You can find more information on multitasking support and CICS support in *OS PL/I Optimizing Compiler: Programmer's Guide.*

Before installing the compiler or the composite product, you must decide whether you want TSO support at your site. You can find more information on TSO support in *OS PL/I Optimizing Compiler: TSO User's Guide.*

You will use SMP control statements to choose or exclude these optional subcomponents during the installation process.

Choosing Target Libraries

The data sets into which you will install the executable OS PL/I code are called target libraries. Before allocating these data sets during the installation process, you must decide whether to use private libraries, system libraries, or a combination of the two. All the installation jobs provided on the distribution tape(s) refer to private libraries. If you choose to use system libraries, you must change the jobs to refer to the system libraries you have selected.

We strongly recommend that, at least initially, you use private libraries as your target libraries, for the following reasons:

- By installing the new release into private libraries, you can keep an earlier release as your active PL/I product while installing and verifying the current release. This is particularly important when installing release 5.1, for which the SMP ++VER statement causes the deletion of any existing PL/I libraries unless the new release is installed into private target and SMP data sets. Later, you can make the current release your active product by:

  - Renaming and recataloging the private libraries

  - Moving the new modules from the private libraries to the appropriate system libraries, if you prefer to run PL/I from system libraries

  You must remember to change data set names within your PL/I and SMP procedures when making these changes.

- If your site is using OS PL/I from private libraries at the time you generate a new operating system, it will be relatively easy to move OS PL/I to the new system.

Using Private Libraries

If you decide to keep your OS PL/I product in private libraries, there are certain additional steps you will need to take during and after installation:

- If you choose a private link library such as PLI.PLILINK, you must either add it to the LNKLSTnn member of SYS1.PARMLIB or add STEPLIB DD statements to your cataloged procedures for use at compile and execution time. Figure 15 on page 48 shows a job coded to run the PL/I sample program from private libraries using an in-stream procedure with appropriate STEPLIB DD statements added.

- If you choose a private procedure library such as PLI.PROCLIB, you must use the proper JES start procedure to enable you to use either concatenated or alternate PROCLIBs, or you must use in-stream procedures in the jobs that refer to the cataloged procedures.

- If you choose private resident library data sets such as PLI.PLIBASE or PLI.PLITASK, you must change the OS PL/I procedures' link-edit steps to refer to these names instead of to SYS1.PLIBASE and SYS1.PLITASK.

- If you have selected the compiler's TSO subcomponent, you must also consider the following:

  - If you use a private TSO command processor library such as PLI.CMDLIB, you must concatenate it to SYS1.CMDLIB by using STEPLIB DD statements within your LOGON procedure. If you use a private TSO HELP library such as PLI.PLIHELP, you must concatenate it to SYS1.HELP after the SYSHELP DD statement within your LOGON procedure. Change the LOGON procedure after you install the compiler and before you attempt to use it. Note that private TSO libraries may adversely affect TSO system response time.

  - If you intend to invoke the OS PL/I compiler under TSO EDIT by using the RUN subcommand, you *must* give the resident library's target data set the name SYS1.PLIBASE.

Cataloged Procedures

The distribution tapes provide cataloged procedures to invoke the OS PL/I compiler and libraries, and to install and maintain the product. Both are discussed here.

Product Invocation Cataloged Procedures

The product invocation cataloged procedures (such as PLIXC and PLIXCLG) are generalized for use in many system environments. You can use the SMP USERMOD facility to customize them for your site's system environment. Some of the factors you must consider before modifying the cataloged procedures for your site follow.

You must specify options for the PARM, UNIT, and SPACE parameters of the cataloged procedures, and you must establish the blocking factors for the output data sets. You should choose these according to your site's usual operating procedures.

You must specify a minimum region size for compilation that is at least 8K bytes larger than the largest value that will be specified in the SIZE compiler option, excluding SIZE(MAX). The region size must be at least 128K or larger.

If your system programmer has changed the linkage editor's defaults for "value1" and/or "value2" of the SIZE option, consult your linkage editor and loader user's guide for help in selecting an appropriate REGION parameter. If the linkage editor's default is 256K for "value1" of the SIZE option, the linkage editor region size must be at least 256K in the cataloged procedures PLIXCL, PLIXCLG, and PLIXLG, which use the linkage editor.

If your system programmer has changed the loader's default for the SIZE option, consult your linkage editor and loader user's guide for help in selecting an appropriate REGION parameter. If the loader's default is 300K for the SIZE option, the loader region size must be at least 324K in the cataloged procedures PLIXCG, PLIXG, which use the loader.

Product Installation and Maintenance Cataloged Procedures

The distribution tapes include cataloged procedures that invoke SMP for use during product installation, modification, and maintenance. These procedures are designed to invoke SMP, but contain instructions that enable you to adapt them to invoke SMP/E. These procedures are described below:

L5PROC	You use this procedure in installing the transient library tape and in applying needed USERMODs at the time of installation. If you do not install the resident library or compiler on your system, you will also use this procedure in the future to modify and maintain your transient library.
L4PROC	You use this procedure in installing the resident library tape and in applying needed USERMODs at the time of installation. If you do not install the compiler on your system, you will also use the procedure in the future to modify and maintain both your transient and resident libraries.
P1PROC	You use this procedure in installing the compiler tape and in applying needed USERMODs at the time of installation. If you are installing the transient and resident libraries and the compiler using the three individual tapes, you will also use this procedure in the future to modify and maintain your transient and resident libraries and the compiler.
P3PROC	You use this procedure in installing the composite tape which contains the transient and resident libraries and the compiler. You use it in applying needed USERMODs at the time of installation, and you will also use it in the future to modify and maintain the transient and resident libraries and the compiler.

Changing Option Defaults and Customizing PL/I

After you have installed OS PL/I on your system, it is ready for use. There are a few additional steps you can take at this time to customize your PL/I product for your site's needs.

IBM has supplied sets of default values for compiler options and for execution-time options that are automatically in effect when you invoke PL/I. If you want to change these default values, you can do so either at the time of installation or later using SMP's USERMOD process. General users of OS PL/I can later override either the IBM-supplied default values or those new default values you have established. (Techniques for overriding option default values are documented in *OS PL/I Optimizing Compiler: Programmer's Guide*.) You can also modify the PL/I-supplied error exit routine to issue an abend on abnormal termination of a PL/I program.

Post Installation Tuning

After installing PL/I, you can take certain steps to improve performance and/or reduce storage requirements for PL/I programs.

The shared library facility provides a way to place frequently used resident library modules into an MVS shared library so that multiple PL/I programs executing concurrently in your system can refer to a single copy of the selected resident library modules. This technique can reduce overall storage requirements if a number of PL/I application programs execute concurrently, but can increase slightly the execution time of the PL/I programs.

You can also select frequently used transient library modules to reside in the link pack area. This will assure that these modules are available when needed by executing application programs, and will eliminate the need to load these modules into storage each time they are needed.

Program products such as IMS/VS and CICS give you the ability to specify a list of modules that can be preloaded in order to improve overall system performance. You might find it useful to preload commonly used PL/I library modules.

Note:

SMP does not keep track of modules in the shared library or in the link pack area. Therefore, if you apply to the resident or transient library, or to the CICS interface module DFHSAP, a program temporary fix (PTF) that affects a module in your shared library or in the link pack area, you must rebuild the shared library or recopy the affected module to the link pack area.

Chapter 2. Installing the Composite Product

This chapter describes the procedure for installing the OS PL/I Transient Library, Resident Library, and Optimizing Compiler, as distributed in the composite package, product number 5734-PL3. If you are installing the composite package, you should read this entire chapter before attempting to perform any of the actions described here. If you are installing individual product component tapes, you should begin with Chapter 3, "Installing the Transient Library" on page 23.

This chapter describes the contents of the composite distribution tape and gives the details of the following installation steps:

1. Obtaining the JCL you need from the distribution tape
2. Selecting your target libraries
3. Establishing your site's libraries
4. Installing the compiler and libraries

Distribution Tape Contents

The OS PL/I Transient Library, Resident Library, and Optimizing Compiler, are distributed together on a standard labeled 9-track tape as product number 5734-PL3. The compiler and libraries are also distributed individually on separate tapes as described in subsequent chapters of this book. The *Program Directory* distributed with each product tape describes the files on the tape.

The composite product distribution tape contains the following information:

- SMP control statements and JCLIN
- Selected product source modules, including the:
  - Installation JCL
  - Cataloged procedures to invoke the product
  - Execution-time options module
  - Compiler options modules
  - Macros used by the options modules
  - Shared library macro
  - Sample program
  - Processing Program Table (PPT) entries for the optional CICS component (release 5.1 only)
  - HELP text for the optional TSO component

- Transient library load modules
- Resident library load modules
- Compiler load modules
- Transient and resident library load modules for the optional CICS component (release 5.1 only)
- Transient and resident library load modules for the optional multitasking component
- Compiler load modules for the optional TSO component

Step 1: Obtaining the JCL You Need from the Distribution Tape

The distribution tape contains installation JCL for installing the OS PL/I Optimizing Compiler and Libraries. These jobs are:

P3ALLOC	Allocates the distribution and target libraries.
P3ALSMPE	Allocates and initializes SMP/E data sets.
P3ALSMP4	Allocates and initializes SMP data sets.
P3PROC	Adds an SMP procedure to PLI.PROCLIB.
P3RECEIV	Receives the transient library, resident library, and optimizing compiler product subcomponents you choose.
P3APPLY	Applies the transient library, resident library, and optimizing compiler product subcomponents you choose.
P3MOD	Changes execution-time and compiler option defaults.

The *Program Directory* distributed with the composite tape includes listings of the installation jobs listed above. If you want to use any of these jobs, however, either as they are or with your own modifications, you must copy them from the tape to a data set to which you have access.

Making Your Copy of the Installation Jobs

Figure 1 on page 15 shows a job that invokes the IEBCOPY utility program to copy the installation jobs from the distribution tape to a new temporary data set named PLI.INSTALL.

```
//P3GETJCL JOB MSGLEVEL=(1,1)
//GETJCL  PROC FILENUM=,FMID=
//TAP2DSK EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=A
//*
//* NOTE: DSN, VOL, and LABEL parameters vary with each
//* product release.  Validate the DSN, VOL, and LABEL
//* parameters of the following DD statement by referring
//* to your Program Directory.
//*
//SYSUT1    DD DSN=&FMID..F2,VOL=SER=APL51B,
//             DISP=(SHR,PASS),UNIT=tape,
//             LABEL=(&FILENUM,SL)
//SYSUT2    DD DSN=PLI.INSTALL,DISP=OLD
//SYSUT3    DD UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSUT4    DD UNIT=SYSDA,SPACE=(TRK,(5,1))
//ENDGET  PEND
//ALLOCAT EXEC PGM=IEFBR14
//SAMPJCL   DD DSN=PLI.INSTALL,DISP=(,CATLG),UNIT=SYSDA,
//             VOL=SER=nnnnnn,SPACE=(CYL,(1,1,5))
//*
//* NOTE: FILENUM and FMID parameters vary with each
//* product release.  Validate the FILENUM and FMID
//* parameters of the following EXEC statements by
//* referring to your Program Directory.
//*
//TRANLIB EXEC GETJCL,FILENUM=3,FMID=HTL1510
          COPY INDD=SYSUT1,OUTDD=SYSUT2
        SELECT MEMBER=(IBMBXOPT)
//RESLIB  EXEC GETJCL,FILENUM=11,FMID=HRL1510
          COPY INDD=SYSUT1,OUTDD=SYSUT2
        SELECT MEMBER=(PLIXG,PLIXLG)
//COMPILE EXEC GETJCL,FILENUM=18,FMID=HPL1510
          COPY INDD=SYSUT1,OUTDD=SYSUT2
       EXCLUDE MEMBER=(PLIXOPB)
//TSO     EXEC GETJCL,FILENUM=21,FMID=JPL1514
          COPY INDD=SYSUT1,OUTDD=SYSUT2
        SELECT MEMBER=(IELOAT)
//
```

Figure 1. Job to Copy Installation Jobs from Composite Product Distribution Tape

When you have copied the installation jobs to your PLI.INSTALL data set, you can print them for review or obtain copies that you can modify for use in later steps of the composite product installation process. Figure 2 on page 16 shows a job that invokes the IEBPTPCH utility program to print all the members of the PLI.INSTALL data set.

```
//LISTJCL  JOB MSGLEVEL=(1,1)
//LISTALL EXEC PGM=IEBPTPCH
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSN=PLI.INSTALL,DISP=SHR
//SYSUT2    DD SYSOUT=A
          PRINT TYPORG=PO,MAXFLDS=1
         RECORD FIELD=(80)
//
```

Figure 2. Job to Print All Jobs in the PLI.INSTALL Data Set

In order to use one of the jobs in the PLI.INSTALL data set, you must modify its JOB statement to meet your site's JCL requirements. You might also need to modify other parts of the job. This chapter outlines these modifications.

When your installation is complete and you have ACCEPTed your product, you can erase PLI.INSTALL. ACCEPT processing puts the installation jobs into the distribution library, PLI.PL141S.

Step 2: Selecting Target Libraries

You must choose either private or system libraries into which to install your OS PL/I product. These libraries are referred to as *target libraries*. "Preparing for Installation" on page 8 gives information about system and private libraries and how to choose target libraries. The following table shows the various target libraries you can choose for the OS PL/I product components:

Component	Private Library	System library
Transient library required modules	PLI.PLILINK PLI.PROCLIB	SYS1.LINKLIB SYS1.PROCLIB
Resident library required modules	PLI.PLIBASE PLI.SHRMAC	SYS1.PLIBASE SYS1.SHRMAC
Optimizing compiler required load modules	PLI.PLICOMP	SYS1.LINKLIB
Resident library optional multitasking component	PLI.PLITASK	SYS1.PLITASK
Transient library optional CICS component (release 5.1 only)	PLI.PLILINK	SYS1.LINKLIB
Transient library optional CICS PPT entries (release 5.1 only)	PLI.SAMPLIB	SYS1.SAMPLIB
Resident library optional CICS component (release 5.1 only)	PLI.PLIBASE	SYS1.PLIBASE
Optimizing compiler optional TSO load modules	PLI.CMDLIB	SYS1.CMDLIB

Component	Private Library	System library
Optimizing compiler optional TSO HELP text	PLI.HELP	SYS1.HELP
Sample program	PLI.SAMPLIB	SYS1.SAMPLIB

If you want to use an existing library but adequate space is not available, you can reallocate the existing library using the move/copy utility routine, IEHMOVE.

Note: If you use an existing library, you must remember that, with release 5.1, SMP uses the DELETE operand of the ++VER control statement to *replace* the contents of the existing product data sets.

During the installation process, SMP will link-edit or copy load modules from the distribution tape into your target libraries. It will also copy product invocation cataloged procedures into your PROCLIB target library.

Step 3: Establishing Your Site's Libraries

The installation jobs you copied into your PLI.INSTALL data set include all the jobs necessary to install the transient and resident libraries and the compiler.

Allocating Product Data Sets

You can use the P3ALLOC job from PLI.INSTALL to allocate and catalog the OS PL/I target libraries and distribution libraries.

Space requirements for the OS PL/I data sets are specified in the P3ALLOC installation JCL in terms of blocks. You can use the space allocations as they are shown, or you can convert them to cylinder or track allocations according to the type of DASD device onto which you will install the product. Accurate DASD space requirements for different device types are listed in the *Program Directory* in terms of tracks.

Note:

Before you run the job to allocate storage for the composite product, verify that the total amount of space needed is available on the DASD device you have chosen. In addition, if you plan to use the shared library, you should double the storage allocation for PLI.PLIBASE.

You can find more information on the shared library in "Using the Shared Library" on page 81.

You can obtain the P3ALLOC in-stream procedure using the job shown in Figure 3 on page 18 and customize it for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure or by overriding the defaults in the EXEC statement that invokes the procedure.

```
//PUNCHJCL JOB MSGLEVEL=(1,1)
//PUNCHIT EXEC PGM=IEBGENER
//SYSIN      DD DUMMY
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSN=PLI.INSTALL(P3ALLOC),DISP=SHR
//SYSUT2     DD SYSOUT=B,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//
```

Figure 3. Job to Obtain the P3ALLOC Member of Installation JCL

If you have chosen to use system libraries as your target libraries, you must delete from P3ALLOC the DD statements for the corresponding private libraries. For example, if you have chosen to use SYS1.LINKLIB as your target link library, you must delete the DD statement for PLI.PLILINK; if you have chosen to use SYS1.PROCLIB as your target procedure library, you must delete the DD statement for PLI.PROCLIB, and so on.

If you do not plan to use OS PL/I's multitasking component, you must delete the DD statement for PLITASK. If you do not plan to use OS PL/I's TSO component, you must delete the DD statements for CMDLIB and HELP.

When you have finished modifying P3ALLOC according to your needs, you can execute the P3ALLOC job.

Allocating SMP Data Sets

You can use another installation job from PLI.INSTALL to allocate and initialize your SMP data sets. Use P3ALSMP4 if you use SMP at your site, or P3ALSMPE if you use SMP/E.

You can customize P3ALSMPn for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure or by overriding the defaults in the EXEC statement that invokes the procedure. You must also examine the comment statements in the job to determine additional changes you must make.

Space requirements for the SMP data sets are specified in P3ALSMPn in terms of blocks. You can use the space allocations as they are shown, or you can convert them to cylinder or track allocations according to the type of DASD device onto which you will install the product. Accurate DASD space requirements for different device types are listed in the *Program Directory* in terms of tracks.

Note:

Before you run the job to allocate storage for the SMP data sets, verify that the total amount of space needed is available on the DASD device you have chosen.

You can obtain either P3ALSMP4 or P3ALSMPE by using the PUNCHJCL job shown in Figure 3 on page 18. Replace the name 'P3ALLOC' in the SYSUT1 DD statement with the name of the member that corresponds to the version of SMP used at your site.

Adding the Cataloged Procedure to Install

You can use the P3PROC job from PLI.INSTALL to place the P3PROC cataloged procedure into PLI.PROCLIB. The P3PROC procedure is used by P3RECEIV and P3APPLY, which install the transient and resident libraries and the compiler, and also by P3MOD if you choose to change the IBM-supplied execution-time and/or compiler option defaults.

Before placing P3PROC into PLI.PROCLIB, you can customize it for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure. You can identify, by comments in the P3PROC job, the JCL statements that are specific to SMP or SMP/E; use those statements that match your site's version of SMP.

The data set names in the target library DD statements must match the names you have selected for your site. If you decided to use system libraries as your target libraries, you must change the DSN portions of the corresponding DD statements to specify the appropriate system libraries. If you decided not to use the multitasking component, delete the PLITASK DD statement. If you decided not to use the TSO component, delete the CMDLIB and HELP DD statements.

You can obtain the P3PROC installation job by using the PUNCHJCL job shown in Figure 3 on page 18. Just replace the data set member name in the SYSUT1 DD statement with the name 'P3PROC'.

Step 4: Installing the Distribution Tape

When you have run the jobs to allocate the transient library, resident library, and compiler data sets, allocate and initialize the SMP data sets, and add P3PROC to your procedure library, you are ready to install the distribution tape. You can use the P3RECEIV job from PLI.INSTALL to RECEIVE the OS PL/I Optimizing Compiler composite product, and the P3APPLY job to APPLY it.

If you are using SMP to install, you can find more information about the RECEIVE and APPLY procedures in *OS/VS System Modification Program (SMP) System Programmer's Guide*. If you are using SMP/E to install, you can find more information about the RECEIVE and APPLY procedures in *SMP/E User's Guide*, and *SMP/E Reference*. You also have the option of using the interactive capability of SMP/E to install rather than the batch procedure described here. This is described in *SMP/E Terminal User's Guide*.

RECEIVEing the Composite Product

Job P3RECEIV invokes P3PROC to RECEIVE compiler and library data sets from the distribution tape into temporary data sets allocated by SMP. If you installed P3PROC into private procedure library PLI.PROCLIB, you must use the proper JES start procedure to enable you to use concatenated or alternate PROCLIBs, or insert P3PROC, ended by a // PEND statement, into P3RECEIV as an in-stream procedure.

If you want to override some of P3PROC's symbolic parameters, you can do so with the EXEC PROC statement you use in P3RECEIV to execute P3PROC.

You must also verify the UNIT parameter in the SMPPTFIN DD statement in P3RECEIV. Several DD statements are provided and represent the possible combinations of product function and SMP level. These are identified by the comments accompanying them. You must select the group of input data that matches your OS PL/I product components and your SMP product. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL. The other DD statements in this job will have no effect.

When you have finished modifying P3RECEIV according to your site's needs, you can execute the P3RECEIV job.

APPLYing the Composite Product

Remember to adjust for using a private procedure library by using the appropriate JES start procedure or inserting P3PROC into P3APPLY. You can, if you want to, use the EXEC PROC statement in P3APPLY to override P3PROC's symbolic parameters

P3APPLY provides several DD statements that represent the possible combinations of transient and resident library and optimizing compiler functions and SMP level. You can identify these by the comments that accompany them. You must select the group of input data that matches your PL/I product components and your SMP product. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL. The other DD statements in P3APPLY will have no effect.

When you have finished modifying P3APPLY according to your site's needs, you can execute the P3APPLY job.

Note: During the execution of P3APPLY, you will receive the following messages from the linkage editor:

```
IEW0461 SYMBOL PRINTED IS AN UNRESOLVED EXTERNAL REFERENCE
IEW0342 ERROR - LIBRARY SPECIFIED DOES NOT CONTAIN MODULE
```

IEW0461 is issued during the link-edit of compiler module IEL0PA. You should ignore the unresolved external reference to IEAXPALL because this module is not used in either an MVS or MVS/XA environment.

IEW0461 is also issued during the link-edit of some of the resident library modules. These references will later be resolved when a compiled application program is link-edited.

IEW0342, which sets return code 08, is issued because SMP attempts to include, from the target library, members that do not yet exist. These messages are normal during initial installation.

Verifying Correct Installation

After P3APPLY has executed correctly, the PL/I transient library, resident library, and optimizing compiler are ready for use. You will probably want to run the sample program to test the success of the installation process. See Chapter 6, "Running the Installation Verification Program" on page 47.

ACCEPTing the OS PL/I Product

You will probably want to try out your new compiler and libraries for a while before you execute an SMP ACCEPT command to permanently install the product. You *must*, however, ACCEPT the product before you attempt to install any maintenance or USERMODs, as you will not be able to RESTORE the product to its previous level (in the event of a problem) if you have not previously executed the SMP ACCEPT command against the product itself.

You can easily modify P3APPLY to ACCEPT the product:

- If you are using SMP, just change the APPLY statement to ACCEPT.
- For SMP/E, change the APPLY statement to ACCEPT, and change the target zone designation from PLITGT to PLIDLIB.

Completing Installation of CICS Support

If you have chosen to install PL/I release 5.1's support for CICS, there are some additional steps you must take after ACCEPTing your PL/I products to connect the transient and resident library CICS components to the CICS product. These steps are described in Chapter 7, "Completing Installation of PL/I Support for CICS" on page 51.

Customizing and Tuning Your OS PL/I Product

You can customize your PL/I product by changing the default values supplied by IBM for the execution-time and compiler options. You can also modify the PL/I error exit routine to issue an abend on abnormal termination of a PL/I program. Chapter 8, "Changing Option Defaults and Customizing PL/I" on page 55 describes the options and their IBM-supplied default values, and explains how to change the defaults according to the needs of your site. It also describes how to modify the error exit routine.

You can also use the shared library, add transient library modules to the link pack area, and disable PL/I's fast initialization facility if storage for PL/I program load modules is limited. These procedures are discussed in Chapter 9, "Postinstallation Tuning" on page 81.

Chapter 3. Installing the Transient Library

This chapter describes the procedure for installing the OS PL/I Transient Library. If you are installing the individual transient library tape, you should read this entire chapter before attempting to perform any of the actions described here. If you are installing the composite tape that contains the transient and resident libraries and the compiler, you should use the procedures described in Chapter 2, "Installing the Composite Product" on page 13 to install the product.

This chapter describes the contents of the distribution tape and gives the details of the following installation steps:

1. Obtaining the JCL you need from the distribution tape
2. Selecting your target libraries
3. Establishing your site's libraries
4. Installing the transient library

Distribution Tape Contents

The OS PL/I Transient Library is distributed on a standard labeled 9-track tape as product number 5734-LM5 or as part of the composite product number 5734-PL3, which also includes the Resident Library and the Optimizing Compiler. The *Program Directory* distributed with each product tape describes the files on the tape.

The Transient Library distribution tape contains the following information:

- SMP control statements and JCLIN
- Selected transient library source modules, including the:
  - Installation JCL
  - Execution-time options module
  - Macro used by the options module
  - Processing Program Table (PPT) entries for the optional CICS component (release 5.1 only)
- Transient library load modules

- Transient library load modules for the optional CICS component (release 5.1 only)
- Transient library load modules for the optional multitasking component

Step 1: Obtaining the JCL You Need from the Distribution Tape

The distribution tape provides JCL for installing the transient library. The job names and their descriptions follow:

L5ALLOC	Allocates the distribution and target libraries.
L5ALSMPE	Allocates and initializes SMP/E data sets.
L5ALSMP4	Allocates and initializes SMP data sets.
L5PROC	Adds an SMP procedure to PLI.PROCLIB.
L5RECEIV	Receives the transient library product subcomponents you choose.
L5APPLY	Applies the transient library product subcomponents you choose.
L5MOD	Changes execution-time options defaults.

The *Program Directory* distributed with the product tape includes listings of the installation jobs listed above. If you want to use any of the jobs, however, either as they are or with your own modifications, you must copy them to a data set to which you have access.

Making Your Copy of the Installation Jobs

Figure 4 on page 25 shows a job that invokes the IEBCOPY utility program to copy the installation jobs from the distribution tape to a new temporary data set named PLI.INSTALL.

```
//L5GETJCL JOB MSGLEVEL=(1,1)
//TAP2DSK EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=A
//*
//* NOTE: DSN and VOL parameters vary with each product
//* release.  Validate the DSN and VOL parameters of the
//* following DD statement in your Program Directory.
//*
//SYSUT1     DD DSN=HTL1510.F2,VOL=SER=ATL51A,
//              DISP=SHR,UNIT=tape,LABEL=(3,SL)
//SYSUT2     DD DSN=PLI.INSTALL,DISP=(,CATLG),UNIT=SYSDA,
//              VOL=SER=nnnnnn,SPACE=(CYL,(1,1,5))
//SYSUT3     DD UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSUT4     DD UNIT=SYSDA,SPACE=(TRK,(5,1))
           COPY INDD=SYSUT1,OUTDD=SYSUT2
        EXCLUDE MEMBER=(PLTRLIB)
//
```

Figure 4. Job to Copy Installation Jobs from the Transient Library Tape

When you have copied the installation jobs to your PLI.INSTALL data set, you can print them for review or obtain copies that you can modify for use in later steps of the transient library installation process. Figure 5 shows a job that invokes the IEBPTPCH utility program to print all the jobs in the PLI.INSTALL data set.

```
//LISTJCL   JOB MSGLEVEL=(1,1)
//LISTALL EXEC PGM=IEBPTPCH
//SYSPRINT  DD SYSOUT=A
//SYSUT1     DD DSN=PLI.INSTALL,DISP=SHR
//SYSUT2     DD SYSOUT=A
          PRINT TYPORG=PO,MAXFLDS=1
         RECORD FIELD=(80)
//
```

Figure 5. Job to Print All the Jobs in the PLI.INSTALL Data Set

In order to use one of the jobs in the PLI.INSTALL data set, you must modify its JOB statement to meet your site's JCL requirements. You might also need to modify other parts of the job. Such modifications are described in the remainder of this chapter.

When your installation is complete and you have ACCEPTed your product, you can erase PLI.INSTALL. ACCEPT processing puts the installation jobs into the distribution library, PLI.LM541S.

Step 2: Selecting Target Libraries

You must select either private or system link and procedure libraries into which you will install the transient library. These libraries are referred to as *target libraries.* See "Preparing for Installation" on page 8 for more information about system and private libraries and how to choose target libraries. The following table shows the various target libraries you can choose for the transient library:

Component	Private Library	System library
Transient library required modules	PLI.PLILINK PLI.PROCLIB	SYS1.LINKLIB SYS1.PROCLIB
Transient library optional multitasking component	PLI.PLILINK	SYS1.LINKLIB
Transient library optional CICS component (release 5.1 only)	PLI.PLILINK	SYS1.LINKLIB
Transient library optional CICS PPT entries(release 5.1 only)	PLI.SAMPLIB	SYS1.SAMPLIB

If you want to use an existing library but adequate space is not available, you can reallocate the existing library, using the move/copy utility routine, IEHMOVE.

Note: If you are installing into an existing library, you must remember that, with release 5.1, SMP uses the DELETE operand of the ++VER control statement to *replace* the contents of the existing product data set.

During the installation process, SMP will link-edit or copy load modules from your distribution tape into your target libraries.

Step 3: Establishing Your Site's Libraries

The installation jobs you copied into your PLI.INSTALL data set include all the jobs necessary to install the transient library.

Allocating Transient Library Data Sets

You can use the L5ALLOC job from PLI.INSTALL to allocate and catalog the OS PL/I Transient Library target libraries and distribution libraries.

Space requirements for the transient library modules are specified in the L5ALLOC installation job in terms of blocks. You can use the space allocations as they are shown, or you can convert them to cylinder or track allocations according to the type of DASD device onto which you will install the transient library. Accurate DASD space requirements for different device types are listed in the *Program Directory* in terms of tracks.

Note:

Before you run the job to allocate storage for the transient library, verify that the total amount of space needed is available on the DASD device you have chosen.

You can obtain the L5ALLOC in-stream procedure using the job shown in Figure 6, then customize it for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure or by overriding the defaults in the EXEC statement that invokes the procedure. If you have chosen to use SYS1.LINKLIB as your target link library, you must delete the DD statement for PLI.PLILINK before executing L5ALLOC. If you have chosen to use SYS1.PROCLIB as your target procedure library, you must delete the DD statement for PLI.PROCLIB before executing L5ALLOC.

If you are installing the PL/I release 5.1 support for CICS, we strongly recommend that you install the transient library CICS load modules into the private library PLI.PLILINK, and the CICS PPT entries into the private library PLI.SAMPLIB. The procedures for enabling CICS to communicate with PL/I are discussed in Chapter 7, "Completing Installation of PL/I Support for CICS" on page 51. If you choose, however, to install the CICS modules into system rather than private libraries, you must delete the DD statements for PLI.PLILINK and PLI.SAMPLIB.

When you have finished modifying L5ALLOC according to your needs, you can execute the L5ALLOC job.

```
//PUNCHJCL JOB MSGLEVEL=(1,1)
//PUNCHIT EXEC PGM=IEBGENER
//SYSIN      DD DUMMY
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSN=PLI.INSTALL(L5ALLOC),DISP=SHR
//SYSUT2     DD SYSOUT=B,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//
```

Figure 6. Job to Obtain L5ALLOC Installation Job

Allocating SMP Data Sets

You can use another installation job from PLI.INSTALL to allocate and initialize your SMP data sets. Use L5ALSMP4 if you use SMP at your site, or L5ALSMPE if you use SMP/E.

You can customize the L5ALSMPn in-stream procedure for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure or by overriding the defaults in the EXEC statement that invokes the procedure. Read all the comments in L5ALSMPn to determine the other changes you must make to the job.

Space requirements for the SMP data sets are specified in the L5ALSMPn installation job in terms of blocks. You can use the space allocations as they are shown, or you can convert them to cylinder or track allocations according to the type of DASD device onto which you will install the libraries. Accurate DASD space requirements for different device types are listed in the *Program Directory*.

Note:

Before you run the job to allocate storage for the SMP data sets, verify that the total amount of space needed is available on the DASD device you have chosen.

You can obtain either L5ALSMP4 or L5ALSMPE by using the PUNCHJCL job shown in Figure 6 on page 27. Just replace the name 'L5ALLOC' in the SYSUT1 DD statement with the name of the job that corresponds to the version of SMP used at your site.

Adding the Cataloged Procedure to Install

You can use the L5PROC job in PLI.INSTALL to place the L5PROC cataloged procedure into PLI.PROCLIB. The L5PROC procedure is used by L5RECEIV and L5APPLY, which install the transient library, and also by L5MOD, if you choose to change the IBM-supplied execution-time option defaults.

You can customize the L5PROC job for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure. You can identify, by comments in the job, the L5PROC JCL statements that are specific to SMP or SMP/E; use those statements that match your site's version of SMP. The target library name in the PLILINK DD statement must match the name you have selected for your site.

If you chose the system link library (SYS1.LINKLIB) as your target link library, you must change the DSN portion of the PLILINK statement to specify SYS1.LINKLIB. If you chose the system procedure library (SYS1.PROCLIB) as your target procedure library, you must change all occurrences of PLI.PROCLIB to specify SYS1.PROCLIB.

You can obtain the L5PROC installation job by using the PUNCHJCL job shown in Figure 6 on page 27. Just replace the data set member name in the SYSUT1 DD statement with the name 'L5PROC'.

Step 4: Installing the Distribution Tape

When you have run the jobs to allocate the transient library data sets, to allocate and initialize the SMP data sets, and to place L5PROC in your procedure library, you are ready to install the distribution tape. You can use the L5RECEIV job from PLI.INSTALL to RECEIVE the OS PL/I Transient Library product, and the L5APPLY job to APPLY it.

If you are using SMP to install, you can find more information about the RECEIVE and APPLY procedures in *OS/VS System Modification Program (SMP) System Programmer's Guide.* If you are using SMP/E to install, you can find more information about the RECEIVE and APPLY procedures in *SMP/E User's Guide*, and *SMP/E Reference.* You also have the option of using the interactive capability of SMP/E to install rather than the batch procedure described here. This is described in *SMP/E Terminal User's Guide.*

RECEIVEing the Transient Library

Job L5RECEIV invokes L5PROC to RECEIVE transient library data sets from the distribution tape into temporary data sets allocated by SMP. If you installed L5PROC into private procedure library PLI.PROCLIB, you must use the proper JES start procedure to enable you to use concatenated or alternate PROCLIBs, or insert L5PROC, ended by a // PEND statement, into L5RECEIV as an in-stream procedure.

If you want to override some of L5PROC's symbolic parameters, you can do so with the EXEC PROC statement you use in L5RECEIV to execute L5PROC.

You must also verify the UNIT parameter in the SMPPTFIN DD statement in L5RECEIV. Several DD statements are provided and represent the possible combinations of transient library function and SMP level. These are identified by the comments accompanying them. You must select the grouping of input data that matches the transient library component(s) you have chosen and your SMP product. Change the DDNAME on the single matching //SMPnnn DD * statement in L5RECEIV to SMPCNTL. The other DD statements in this job will have no effect.

When you have finished modifying L5RECEIV according to your site's needs, you can execute the L5RECEIV job.

APPLYing the Transient Library

Note:

If you are also installing the resident library or the resident library and optimizing compiler, *do not* run L5APPLY. Instead, turn now to Chapter 4, "Installing the Resident Library" on page 31. If you are installing *only* the transient library, continue with the remaining instructions in this chapter.

Remember to adjust for using a private procedure library by using the appropriate JES start procedure or by inserting L5PROC into L5APPLY. You can, if you wish, use the EXEC PROC statement in L5APPLY to override L5PROC's symbolic parameters.

L5APPLY provides several DD statements that represent the possible combinations of transient library function and SMP level. You can identify these by the comments that accompany them. You must select the group of input data that matches the transient library components you have chosen and your SMP product. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL. The other DD statements in L5APPLY will have no effect. When you have finished modifying L5APPLY according to your site's needs, you can execute the L5APPLY job.

Note: During execution of L5APPLY, you will receive the following message from the linkage editor:

```
IEW0342 ERROR - LIBRARY SPECIFIED DOES NOT CONTAIN MODULE
```

This message, which sets return code 08, is issued because SMP attempts to include, from the target library, members that do not yet exist. This message is normal during initial installation.

ACCEPTing the Transient Library

You will probably want to try out your new transient library for a while before you execute an SMP ACCEPT command to permanently install the product. You must, however, ACCEPT the product before you attempt to install any maintenance or USERMODs, as you will not be able to RESTORE the product to its previous level (in the event of a problem) if you have not previously executed the ACCEPT command against the product itself.

You can easily modify L5APPLY to ACCEPT the product:

- If you are using SMP, change the APPLY statement to ACCEPT.
- If you are using SMP/E, change the APPLY statement to ACCEPT and change the target zone designation from PLITGT to PLIDLIB.

Completing Installation of CICS Support

If you have chosen to install PL/I release 5.1's support for CICS, there are some additional steps you must take after ACCEPTing your PL/I transient library to connect its CICS component to the CICS product. These steps are described in Chapter 7, "Completing Installation of PL/I Support for CICS" on page 51.

Customizing and Tuning Your OS PL/I Product

You can customize your transient library by changing the default values supplied by IBM for the execution-time options. Chapter 8, "Changing Option Defaults and Customizing PL/I" on page 55 describes the options and their IBM-supplied default values, and explains how to change the defaults according to the needs of your site.

You can also add transient library modules to the link pack area and disable PL/I's fast initialization facility if storage for PL/I program load modules is limited. These procedures are discussed in Chapter 9, "Postinstallation Tuning" on page 81.

Chapter 4. Installing the Resident Library

This chapter describes the procedure for installing the OS PL/I Resident Library. If you are installing the individual resident library tape, you should read this entire chapter before attempting to perform any of the actions described here. If you are installing the composite tape that includes the transient library and compiler, you should use the procedures outlined in Chapter 2, "Installing the Composite Product" on page 13 to install the product.

This chapter describes the contents of the distribution tape and gives the details of the following installation steps:

1. Obtaining the JCL you need from the distribution tape
2. Selecting your target libraries
3. Establishing your site's libraries
4. Installing the resident library

Note:

You *must* run the transient library installation jobs through the L5RECEIV job before you begin installing the resident library. If you have previously installed the transient library (that is, you have run the L5APPLY job), and you now want to install the resident library, you must *first reinstall the transient library through the L5RECEIV step before you install the resident library.*

Distribution Tape Contents

The OS PL/I Resident Library is distributed on a standard labeled 9-track tape as product number 5734-LM4 or as part of the composite product number 5734-PL3, which includes the Transient Library and the Optimizing Compiler. The *Program Directory* distributed with each product tape describes the files on the tape.

The Resident Library distribution tape contains the following information:

- SMP control statements and JCLIN
- Selected resident library source modules, including the:
  - Installation JCL
  - Cataloged procedures to invoke the product
  - Shared library macro
- Resident library load modules
- Resident library load modules for the optional CICS component (release 5.1 only)
- Resident library load modules for the optional multitasking component

Step 1: Obtaining the JCL You Need from the Distribution Tape

The distribution tape provides JCL for installing the resident library. The job names and their descriptions follow:

L4ALLOC	Allocates the distribution and target libraries.
L4PROC	Adds an SMP procedure to PLI.PROCLIB.
L4RECEIV	Receives the resident library product components you choose.
L4APPLY	Applies the transient library and resident library products you have chosen.

The *Program Directory* distributed with the product tape includes listings of the installation jobs listed above. If you want to use any of the jobs, however, either as they are or with your own modifications, you must copy them to a data set to which you have access.

Making Your Copy of the Installation Jobs

Figure 7 on page 33 shows a job that invokes the IEBCOPY utility program to copy the installation jobs from the distribution tape to the existing temporary data set named PLI.INSTALL, which you created during the installation of the transient library.

```
//L4GETJCL JOB MSGLEVEL=(1,1)
//TAP2DSK EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=A
//*
//* NOTE: DSN and VOL parameters vary with each  product
//* release.  Validate the DSN and VOL parameters of the
//* following DD statement in your Program Directory.
//*
//SYSUT1     DD DSN=HRL1510.F2,VOL=SER=ARL51A,
//              DISP=SHR,UNIT=tape,LABEL=(3,SL)
//SYSUT2     DD DSN=PLI.INSTALL,DISP=OLD
//SYSUT3     DD UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSUT4     DD UNIT=SYSDA,SPACE=(TRK,(5,1))
           COPY INDD=SYSUT1,OUTDD=SYSUT2
        EXCLUDE MEMBER=(PLRSHR)
//
```

Figure 7. Job to Copy Installation Jobs from the Resident Library Tape

When you have copied the installation jobs to your PLI.INSTALL data set, you can print them for review or obtain copies that you can modify for use in later steps of the resident library installation process. Figure 8 shows a job that invokes the IEBPTPCH utility program to print all the jobs in the PLI.INSTALL data set.

```
//LISTJCL   JOB MSGLEVEL=(1,1)
//LISTALL EXEC PGM=IEBPTPCH
//SYSPRINT  DD SYSOUT=A
//SYSUT1     DD DSN=PLI.INSTALL,DISP=SHR
//SYSUT2     DD SYSOUT=A
          PRINT TYPORG=PO,MAXFLDS=1
         RECORD FIELD=(80)
//
```

Figure 8. Job to Print All the Jobs in the PLI.INSTALL Data Set

In order to use one of the jobs in the PLI.INSTALL data set, you must modify its JOB statement to meet your site's requirements. You might also need to modify other parts of the job. Such modifications are described in the remainder of this chapter.

When your installation is complete and you have ACCEPTed your product, you can erase PLI.INSTALL. ACCEPT processing puts the installation jobs into the distribution library, PLI.LM441S.

Step 2: Selecting Target Libraries

You must select either a private or system library as a target library to which you will install the resident library. (See "Preparing for Installation" on page 8 for more information about system and private libraries and how to choose a target library.) The following table shows the various target libraries you can choose for the resident library:

Component	Private Library	System library
Resident library required modules	PLI.PLIBASE PLI.SHRMAC	SYS1.PLIBASE SYS1.SHRMAC
Resident library optional multitasking component	PLI.PLITASK	SYS1.PLITASK
Resident library optional CICS component (release 5.1 only)	PLI.PLIBASE	SYS1.PLIBASE

If you want to use an existing library but adequate space is not available, you can reallocate the existing library using the move/copy utility routine, IEHMOVE.

Note: If you use an existing library, you must remember that, with release 5.1, SMP uses the DELETE operand of the ++VER control statement to *replace* the contents of the existing product data set.

During the installation process, SMP will link-edit or copy load modules from your distribution tape into your target libraries. It will also copy product invocation cataloged procedures into the PROCLIB target library you selected during transient library installation.

If you require multitasking support and have installed the multitasking component of the transient library, you must install the multitasking component of the resident library into your PLITASK target library.

If you require CICS support and have installed the release 5.1 CICS component of the transient library, you must install the CICS component of the resident library into your PLIBASE target library.

Step 3: Establishing Your Site's Libraries

The installation jobs you copied into your PLI.INSTALL data set include all the jobs necessary to install the resident library.

Allocating Resident Library Data Sets

You can use the L4ALLOC job from PLI.INSTALL to allocate and catalog the OS PL/I Resident Library target libraries and distribution libraries.

Space requirements for the resident library modules are specified in the L4ALLOC installation JCL in terms of blocks. You can use the space allocations as they are shown, or you can convert them to cylinder or track allocations according to the type of DASD device onto which you are installing the libraries. Accurate DASD space requirements for different device types are listed in the *Program Directory* in terms of tracks.

Note:

Before you run the job to allocate storage for the resident library, verify that the total amount of space needed is available on the DASD device you have chosen. In addition, if you plan to use the shared library, you should double the storage allocation for PLI.PLIBASE.

You can find more information on the shared library in "Using the Shared Library" on page 81.

You can obtain the L4ALLOC in-stream procedure using the job shown in Figure 9, then customize it for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure or by overriding them in the EXEC statement that invokes the procedure.

If you do not plan to use OS PL/I's multitasking component, you must delete the DD statement for PLITASK.

When you have finished modifying L4ALLOC according to your needs, you can execute L4ALLOC.

```
//PUNCHJCL JOB MSGLEVEL=(1,1)
//PUNCHIT EXEC PGM=IEBGENER
//SYSIN      DD DUMMY
//SYSPRINT   DD SYSOUT=A
//SYSUT1     DD DSN=PLI.INSTALL(L4ALLOC),DISP=SHR
//SYSUT2     DD SYSOUT=B,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//
```

Figure 9. Job to Obtain L4ALLOC Installation Job

Adding the Cataloged Procedure to Install

You can use the L4PROC job from PLI.INSTALL to place the L4PROC cataloged procedure into PLI.PROCLIB. The L4PROC procedure is used by L4RECEIV and L4APPLY, which install the resident library.

Before placing L4PROC in PLI.PROCLIB, you can customize it for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure. You can identify, by comments in the job, the L4PROC JCL statements that are specific to SMP or SMP/E; use those statements which match your site's version of SMP. The data set names in the target library DD statements must match the names you have selected for your site.

If you decided to use system libraries as your target libraries, you must change the DSN portions of the corresponding DD statements to specify the appropriate system libraries. If you decided not to use multitasking, delete the DD statement for PLITASK.

You can obtain the L4PROC installation job by using the PUNCHJCL job shown in Figure 9. Just replace the data set member name in the SYSUT1 DD statement with the name 'L4PROC'.

Step 4: Installing the Distribution Tape

When you have run the jobs to allocate the resident library data sets and add L4PROC to your procedure library, you are ready to install the distribution tape. You can use the L4RECEIV job from PLI.INSTALL to RECEIVE the OS PL/I Resident Library product and the L4APPLY job to apply it and the transient library.

If you are using SMP to install, you can find more information about the RECEIVE and APPLY procedures in *OS/VS SMP Programmer's Guide.* If you are using SMP/E to install, you can find more information about the RECEIVE and APPLY procedures in *SMP/E User's Guide* and *SMP/E Reference.* You also have the option of using the interactive capability of SMP/E to install rather than the batch procedure described here. This is described in *SMP/E Terminal User's Guide.*

RECEIVEing the Resident Library

Job L4RECEIV invokes L4PROC to RECEIVE resident library data sets from the distribution tape into temporary data sets allocated by SMP. If you installed L4PROC into private procedure library PLI.PROCLIB, you must use the proper JES start procedure to enable you to use concatenated or alternate PROCLIBs, or insert L4PROC, ended by a // PEND statement, into L4RECEIV as an in-stream procedure.

If you want to override some of L4PROC's symbolic parameters, you can do so with the EXEC PROC statement you use in L4RECEIV to execute L4PROC.

You must also verify the UNIT parameter in the SMPPTFIN DD statement. Several DD statements are provided and represent the possible combinations of resident library function and SMP level. These are identified by the comments accompanying them. You must select the group of input data that matches your resident library component(s) and your SMP product. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL. The other DD statements in the job will have no effect.

When you have finished modifying L4RECEIV according to your site's needs, you can execute the L4RECEIV job.

APPLYing the Transient and Resident Libraries

Note:

If you are also installing the optimizing compiler using the individual product tape, *do not* run L4APPLY. Instead, turn now to Chapter 5, "Installing the Optimizing Compiler" on page 39. If you are installing *only* the transient library and the resident library, continue with the remaining instructions in this chapter.

Remember to adjust for using a private procedure library by using the appropriate JES start procedure or inserting L4PROC into L4APPLY. You can, if you wish, use the EXEC PROC statement in L4APPLY to override L4PROC's symbolic parameters. L4APPLY provides several DD statements that represent the possible combinations of transient and resident library function and SMP level. You can identify these by the comments that accompany them. You must select the group of input data that matches your transient and resident library components and your SMP product. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL. The other DD statements in L4APPLY will have no effect.

When you have finished modifying L4APPLY according to your site's needs, you can execute the L4APPLY job.

Note: During the execution of L4APPLY, you will receive the following messages from the linkage editor:

```
IEW0461 SYMBOL PRINTED IS AN UNRESOLVED EXTERNAL REFERENCE
IEW0342 ERROR - LIBRARY SPECIFIED DOES NOT CONTAIN MODULE
```

IEW0461 is issued during the link-edit of some of the resident library modules. These unresolved references will later be resolved when a compiled application program is link-edited.

IEW0342, which sets return code 08, is issued because SMP attempts to include, from the target library, members that do not yet exist. These messages are normal during initial installation.

ACCEPTing the Transient and Resident Library

You will probably want to try out your new transient and resident libraries for a while before you execute an SMP ACCEPT command to permanently install the product. You must, however, ACCEPT the product before you attempt to install any maintenance or USERMODs, as you will not be able to RESTORE the product to its previous level (in the event of a problem) if you have not previously executed the ACCEPT command against the product itself.

You can easily modify L4APPLY to ACCEPT the product:

- If you are using SMP, change the APPLY statement to ACCEPT.

- If you are using SMP/E, change the APPLY statement to ACCEPT and change the target zone designation from PLITGT to PLIDLIB.

Completing Installation of CICS Support

If you have chosen to install PL/I release 5.1's support for CICS, there are some additional steps you must take after ACCEPTing your PL/I transient and resident libraries to connect their CICS components to the CICS product. These steps are described in Chapter 7, "Completing Installation of PL/I Support for CICS" on page 51.

Customizing and Tuning Your OS PL/I Product

You can customize your transient library product by changing the default values supplied by IBM for the execution-time options. Chapter 8, "Changing Option Defaults and Customizing PL/I" on page 55, describes the options and their IBM-supplied default values, and explains how to change the defaults according to the needs of your site. It also describes how to modify the PL/I-supplied error exit routine to issue and abend on abnormal termination of a PL/I program.

You can also use the shared library, add transient library modules to the link pack area, and disable PL/I's fast initialization facility if storage for PL/I program load modules is limited. These procedures are discussed in Chapter 9, "Postinstallation Tuning" on page 81.

Chapter 5. Installing the Optimizing Compiler

This chapter describes the procedure for installing the OS PL/I Optimizing Compiler. If you are installing the individual compiler tape, you should read this entire chapter before attempting to perform any of the actions described here. If you are installing the composite tape that contains the compiler and the transient and resident libraries, you should use the procedures outlined in Chapter 2, "Installing the Composite Product" on page 13 to install the product.

This chapter describes the contents of the distribution tape and gives the details of the following installation steps:

1. Obtaining the JCL you need from the distribution tape

2. Selecting your target libraries

3. Establishing your site's libraries

4. Installing the compiler

Note:

You *must* run the transient library installation jobs through the L5RECEIV job and the resident library installation jobs through the L4RECEIV job before you begin installing the optimizing compiler. If you have previously installed the transient and resident libraries (that is, if you have run the L4APPLY job), and you now want to install the optimizing compiler, you must first *reinstall the transient and resident libraries.*

Distribution Tape Contents

The OS PL/I Optimizing Compiler is distributed on a standard labeled 9-track tape as product number 5734-PL1 or as part of the composite product number 5734-PL3, which includes the Transient Library and the Resident Library. The *Program Directory* distributed with each product tape describes the files on the tape.

The Optimizing Compiler distribution tape contains the following information:

- SMP control statements and JCLIN

- Selected compiler source modules, including the:
  - Installation JCL
  - Cataloged procedures to invoke the product
  - Compiler options modules
  - Macros used by the options modules
  - Sample program
  - HELP text for the optional TSO component
- Compiler load modules
- Compiler load modules for the optional TSO component

Step 1: Obtaining the JCL You Need from the Distribution Tape

The distribution tape provides JCL for installing the compiler. The job names and their descriptions follow:

P1ALLOC	Allocates the distribution and target libraries.
P1PROC	Adds an SMP procedure to PLI.PROCLIB.
P1RECEIV	Receives the optimizing compiler product components you choose.
P1APPLY	Applies the transient and resident libraries and optimizing compiler product components you have chosen.
P1MOD	Changes compiler options defaults.

The *Program Directory* distributed with the product tape includes listings of the installation jobs listed above. If you want to use any of the jobs, however, either as they are or with your own modifications, you must copy them to a data set to which you have access.

Making Your Copy of the Installation JCL

Figure 10 on page 41 shows a job that invokes the IEBCOPY utility program to copy the installation jobs from the distribution tape to the existing data set named PLI.INSTALL, which you created during the installation of the transient library.

```
//P1GETJCL JOB MSGLEVEL=(1,1)
//GETJCL  PROC FILENUM=,FMID=
//TAP2DSK EXEC PGM=IEBCOPY
//SYSPRINT  DD SYSOUT=A
//*
//* NOTE: DSN, VOL, and LABEL parameters vary with each
//* product release.  Validate the DSN, VOL, and LABEL
//* parameters of the following DD statement by referring
//* to your Program Directory.
//*
//SYSUT1    DD DSN=&FMID..F2,VOL=SER=APL51A,
//             DISP=(SHR,PASS),UNIT=tape,
//             LABEL=(&FILENUM,SL)
//SYSUT2    DD DSN=PLI.INSTALL,DISP=OLD
//SYSUT3    DD UNIT=SYSDA,SPACE=(TRK,(5,1))
//SYSUT4    DD UNIT=SYSDA,SPACE=(TRK,(5,1))
//ENDGET  PEND
//*
//* NOTE: FILENUM and FMID parameters vary with each
//* product release.  Validate the FILENUM and FMID
//* parameters of the following EXEC statements by
//* referring to your Program Directory.
//*
//BASE    EXEC GETJCL,FILENUM=3,FMID=HPL1510
          COPY INDD=SYSUT1,OUTDD=SYSUT2
       EXCLUDE MEMBER=(PLIXOPB)
//TSO     EXEC GETJCL,FILENUM=6,FMID=JPL1514
          COPY INDD=SYSUT1,OUTDD=SYSUT2
        SELECT MEMBER=(IEL0AT)
//
```

Figure 10. Job to Copy Installation Jobs from Optimizing Compiler Tape

When you have copied the installation jobs to your PLI.INSTALL data set, you can print them for review or obtain copies that you can modify for use in later steps of the compiler installation process. Figure 11 shows a job that invokes the IEBPTPCH utility program to print all the jobs in the PLI.INSTALL data set.

```
//LISTJCL  JOB MSGLEVEL=(1,1)
//LISTALL EXEC PGM=IEBPTPCH
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSN=PLI.INSTALL,DISP=SHR
//SYSUT2    DD SYSOUT=A
         PRINT TYPORG=PO,MAXFLDS=1
        RECORD FIELD=(80)
//
```

Figure 11. Job to Print All the Jobs in the PLI.INSTALL Data Set

In order to use one of the jobs in PLI.INSTALL, you must modify its JOB statement to meet your site's requirements. You might also need to modify other parts of the jobs. Such modifications are described in the remainder of this chapter.

When your installation is complete and you have ACCEPTed your product, you can erase PLI.INSTALL. ACCEPT processing puts the installation jobs into the distribution library, PLI.PL141S.

Step 2: Selecting Target Libraries

You must choose either private or system libraries as target libraries into which you will install the optimizing compiler. (See "Preparing for Installation" on page 8 for more information about system and private libraries and how to choose a target library.) The following table shows the various libraries you can choose for the subcomponents of the compiler:

Component	Private Library	System library
Optimizing compiler required load modules	PLI.PLICOMP	SYS1.LINKLIB
Optimizing compiler optional TSO load modules	PLI.CMDLIB	SYS1.CMDLIB
Optimizing compiler optional TSO HELP text	PLI.HELP	SYS1.HELP
Sample program	PLI.SAMPLIB	SYS1.SAMPLIB

If you want to use an existing library but adequate space is not available, you can reallocate the existing library using the move/copy utility routine, IEHMOVE.

Note: If you use an existing library, you must remember that, with release 5.1, SMP uses the DELETE operand of the ++VER control statement to *replace* the contents of the existing product data set.

During the installation process, SMP will link-edit or copy load modules from the distribution tape into your target libraries. It will also copy product invocation cataloged procedures into the PROCLIB target library you selected during transient library installation.

Step 3: Establishing Your Site's Libraries

The installation jobs you copied into your PLI.INSTALL data set include all the jobs necessary to install the compiler.

Allocating Compiler Data Sets

You can use the P1ALLOC job from PLI.INSTALL to allocate and catalog the OS PL/I Optimizing Compiler target libraries and distribution libraries.

Space requirements for the compiler are specified in P1ALLOC in terms of blocks. You can use the space allocations as they are shown, or you can convert them to cylinder or track allocations according to the type of DASD device onto which you will install the compiler. Accurate DASD space requirements for different device types are listed in the *Program Directory* in terms of tracks.

Note:

Before you run the job to allocate storage for the compiler, verify that the total amount of space needed is available on the DASD device you have chosen.

You can obtain the P1ALLOC in-stream procedure using the job shown in Figure 12, then customize it for your site's needs by changing symbolic parameter defaults in the PROC statement of the procedure or by overriding them in the EXEC statement that invokes the procedure. If you have chosen to use system libraries as your target libraries, you must delete the DD statements for the corresponding private libraries before executing P1ALLOC. If you do not plan to use OS PL/I's TSO component, you must delete the DD statements for CMDLIB and HELP.

When you have finished modifying P1ALLOC according to your needs, you can execute P1ALLOC.

```
//PUNCHJCL JOB MSGLEVEL=(1,1)
//PUNCHIT EXEC PGM=IEBGENER
//SYSIN     DD DUMMY
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSN=PLI.INSTALL(P1ALLOC),DISP=SHR
//SYSUT2    DD SYSOUT=B,DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//
```

Figure 12. Job to Obtain P1ALLOC Installation Job

Adding the Cataloged Procedure to Install

You can use the P1PROC job from PLI.INSTALL to place the P1PROC cataloged procedure into PLI.PROCLIB. The P1PROC procedure is used by P1RECEIV and P1APPLY, which install the compiler, and also by P1MOD if you decide to change the IBM-supplied compiler option defaults.

Before placing P1PROC into PLI.PROCLIB, you can customize it for your site's needs by changing symbolic parameter defaults in the PROC statement of the P1PROC procedure. You can identify, by comments in the P1PROC job, the JCL statements that are specific to SMP or SMP/E; use those statements that match your site's version of SMP.

The data set names in the target library DD statements must match the names you have selected for your site. If you decided to use system libraries as your target libraries, you must change the DSN portions of the corresponding DD statements to specify the appropriate system libraries.

If you decided not to use the multitasking component, delete the PLITASK DD statement. If you decided not to use the TSO component, delete the CMDLIB and HELP DD statements.

You can obtain the P1PROC installation job by using the PUNCHJCL job shown in Figure 12 on page 43. Replace the data set member name in the SYSUT1 DD statement with the name 'P1PROC'.

Step 4: Installing the Distribution Tape

When you have run the jobs to allocate the compiler data sets and add P1PROC to your procedure library, you are ready to install the distribution tape. You can use the P1RECEIV member of PLI.INSTALL to RECEIVE the OS PL/I Optimizing Compiler product, and the P1APPLY member to APPLY it and the libraries.

If you are using SMP to install, you can find more information about the RECEIVE and APPLY procedures in *OS/VS SMP Programmer's Guide*. If you are using SMP/E to install, you can find more information about the RECEIVE and APPLY procedures in *SMP/E User's Guide*, and *SMP/E Reference*. You also have the option of using the interactive capability of SMP/E to install rather than the batch procedure described here. This is described in *SMP/E Terminal User's Guide*.

RECEIVEing the Optimizing Compiler

Job P1RECEIV invokes P1PROC to RECEIVE compiler data sets from the distribution tape into temporary data sets allocated by SMP. If you installed P1PROC into private procedure library PLI.PROCLIB, you must use the proper JES start procedure to enable you to use concatenated or alternate PROCLIBs, or insert P1PROC, ended by a // PEND statement, into P1RECEIV as an in-stream procedure.

If you want to override some of P1PROC's symbolic parameters, you can do so with the EXEC PROC statement you use in P1RECEIV to execute P1PROC.

You must also verify the UNIT parameter in the SMPPTFIN DD statement. Several DD statements are provided and represent the possible combinations of compiler components and SMP level. These are identified by the comments accompanying them. You must select the group of input data that matches your compiler components and your SMP product. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL.

When you have finished modifying P1RECEIV according to your site's needs, you can execute the P1RECEIV job.

APPLYing the Transient and Resident Libraries and the Compiler

Remember to adjust for using a private procedure library by using the appropriate JES start procedure or inserting P1PROC into P1APPLY. You can, if you wish, use the EXEC PROC statement in P1APPLY to override P1PROC's symbolic parameters.

P1APPLY provides several DD statements that represent the possible combinations of transient and resident library and optimizing compiler function and SMP level. You can identify these by the comments that accompany them. You must select the group of input data that matches your transient and resident library and optimizing compiler components and your SMP product. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL. The other DD statements in P1APPLY will have no effect.

When you have finished modifying P1APPLY according to your site's needs, you can execute P1APPLY.

Note: During the execution of P1APPLY, you will receive the following messages from the linkage editor:

```
IEW0461 SYMBOL PRINTED IS AN UNRESOLVED EXTERNAL REFERENCE
IEW0342 ERROR - LIBRARY SPECIFIED DOES NOT CONTAIN MODULE
```

IEW0461 is issued during the link-edit of compiler module IEL0PA. You should ignore the unresolved external reference to IEAXPALL because this module is not used in either an MVS or MVS/XA environment.

IEW0461 is also issued during the link-edit of some of the resident library modules. These references will later be resolved when a compiled application program is link-edited.

IEW0342, which sets return code 08, is issued because SMP attempts to include, from the target library, members which do not yet exist. These messages are normal during initial installation.

Verifying Correct Installation

After P1APPLY has executed correctly, the PL/I transient library, resident library, and optimizing compiler are ready for use. You will probably want to run the sample program to test the success of the installation process. See Chapter 6, "Running the Installation Verification Program" on page 47.

ACCEPTing the Compiler

You will probably want to try out your new compiler and libraries for a while before you execute an SMP ACCEPT command to permanently install the product. You must, however, ACCEPT the product before you attempt to install any maintenance or USERMODs, as you will not be able to RESTORE the product to its previous level (in the event of a problem) if you have not previously executed the ACCEPT command against the product itself.

You can easily modify P1APPLY to ACCEPT the product:

- If you are using SMP, change the APPLY statement to ACCEPT
- If you are using SMP/E, change the APPLY statement to ACCEPT and change the target zone designation from PLITGT to PLIDLIB

Completing Installation of CICS Support

If you have chosen to install PL/I release 5.1's support for CICS, there are some additional steps you must take after ACCEPTing your PL/I products to connect the transient and resident library CICS components to the CICS product. These steps are described in Chapter 7, "Completing Installation of PL/I Support for CICS" on page 51.

Customizing and Tuning Your OS PL/I Product

You can customize your PL/I product by changing the default values supplied by IBM for the execution-time and compiler options. You can also modify the PL/I error exit routine to issue an abend on abnormal termination of a PL/I program. Chapter 8, "Changing Option Defaults and Customizing PL/I" on page 55, describes the options and their IBM-supplied default values, and explains how to change the defaults according to the needs of your site. It also describes how to modify the error exit routine.

You can also use the shared library, add transient library modules to the link pack area, and disable PL/I's fast initialization facility if storage for PL/I program load modules is limited. These procedures are discussed in Chapter 9, "Postinstallation Tuning" on page 81.

Chapter 6. Running the Installation Verification Program

The sample program provided on the distribution tape enables you to verify the success of your installation by exercising several of the features of the OS PL/I libraries and compiler. It illustrates the use of the compile-time preprocessor to convert source programs written for the PL/I (F) Compiler to a form appropriate for compilation by the OS PL/I Optimizing Compiler.

You can find a listing of the compilation and execution of the sample program in *OS PL/I Optimizing Compiler Programmer's Guide.*

Printing the Sample Program

Figure 13 shows a job that invokes the IEBPTPCH utility program to print the sample program you installed into the PLI.SAMPLIB data set during SMP APPLY processing. If you chose SYS1.SAMPLIB instead of the private library, change the data set name in the SYSUT1 DD statement accordingly.

```
//LISTSAMP JOB MSGLEVEL=(1,1)
//LISTALL EXEC PGM=IEBPTPCH
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSN=PLI.SAMPLIB,DISP=SHR
//SYSUT2    DD SYSOUT=A
         PRINT TYPORG=PO,MAXNAME=1,MAXFLDS=1
        MEMBER NAME=PLIXSMPB
        RECORD FIELD=(80)
//
```

Figure 13. Job to Print the Sample Program

Compiling, Link-Editing, and Executing the Sample Program

Figure 14 gives an example of JCL you can use to compile, link-edit, and execute the sample program from system libraries.

```
//SAMPLEPL JOB MSGLEVEL=(1,1)
//EXEC    EXEC PROC=PLIXCLG
//PLI.SYSIN DD DSN=SYS1.SAMPLIB(PLIXSMPB),DISP=SHR
```

Figure 14. Example of Job to Run the Sample Program

If you have installed PL/I into private libraries, you must modify the IBM-supplied product invocation cataloged procedures. Figure 15 on page 48 shows a job coded to run the sample program from private libraries using an in-stream PL/I procedure. PLIXCLG is one of the product invocation cataloged procedures provided on the distribution tape. As described in "Using Private Libraries" on page 9, it has been modified on the lines marked with <===NOTE to use private rather than system libraries to compile, link-edit, and execute PL/I.

```
//SAMPLEPL JOB MSGLEVEL=(1,1)
//PLIXCLG  PROC LKLBDSN='PLI.PLIBASE'                               <===NOTE
//PLI      EXEC PGM=IEL0AA,PARM='OBJECT,NODECK',REGION=128K
//STEPLIB  DD DSN=PLI.PLICOMP,DISP=SHR                              <===NOTE
//SYSPRINT DD  SYSOUT=A
//SYSLIN   DD  DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,
//             SPACE=(80,(250,100))
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSDA,
//             SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024
//LKED     EXEC PGM=IEWL,PARM='XREF,LIST',COND=(9,LT,PLI),REGION=256K
//SYSLIB   DD  DSN=&LKLBDSN,DISP=SHR
//         DD  DSN=PLI.PLIBASE,DISP=SHR                             <===NOTE
//SYSLMOD  DD  DSN=&&GOSET(GO),DISP=(MOD,PASS),UNIT=SYSDA,
//             SPACE=(1024,(50,20,1))
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,(200,20)),
//             DCB=BLKSIZE=1024
//SYSPRINT DD  SYSOUT=A
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSIN    DD  DUMMY
//GO       EXEC PGM=*.LKED.SYSLMOD,COND=((9,LT,PLI),(9,LT,LKED)),
//             REGION=100K
//STEPLIB  DD DSN=PLI.PLILINK,DISP=SHR                              <===NOTE
//SYSPRINT DD  SYSOUT=A
//ENDPLIX  PEND
//EXEC     EXEC PROC=PLIXCLG
//PLI.SYSIN DD DSN=PLI.SAMPLIB(PLIXSMPB),DISP=SHR                   <===NOTE
//
```

Figure 15. Job to Run Sample Program from Private Libraries

Interpreting Messages Generated by the Sample Program

During preprocessing and compilation of the sample program, some diagnostic messages are generated. The preprocessor generates warning (W) level messages. The compiler generates error (E), warning (W), and informational (I) messages. One of these is:

```
IEL0413I E  DECLARATION OF 'A' IGNORED
```

IEL0413I E, which sets compiler return code 08, is normal for the compilation of the sample program and will not affect execution.

The execution of the sample program sets return code 0101. This return code is also normal.

ACCEPTing Your PL/I Product

After you have run the sample program and verified that your PL/I product is installed correctly, you should ACCEPT the product. It is especially important to ACCEPT the product before you change any of the IBM-supplied option defaults.

If you installed the composite product 5734-PL3, return to "ACCEPTing the OS PL/I Product" on page 21.

If you installed the individual products 5734-LM5, 5734-LM4, and 5734-PL1, return to "ACCEPTing the Compiler" on page 45.

Chapter 7. Completing Installation of PL/I Support for CICS

If you have installed the optional CICS component of the OS PL/I release 5.1 transient library or of the OS PL/I release 5.1 transient and resident libraries, you must take some additional steps to ensure that CICS can communicate with your new release of PL/I.

First, after ACCEPTing your PL/I product, you must add PL/I entries to the processing program table (PPT) and the destination control table (DCT). Then you must add PL/I CICS data sets to the CICS startup job stream. This chapter describes these procedures.

Adding PPT Entries

The PPT entries listed in the DFHPPT macro statements below are part of the transient library CICS support installed into PLI.SAMPLIB as member PLTCICS during transient library installation.

PL/I support for CICS requires that these entries be added to the CICS processing program table so that they are included when your CICS system is generated.

The PPT macro instructions are:

```
DFHPPT TYPE=ENTRY,PROGRAM=IBMBCCLA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBCCRA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBEOCA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBETAA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBETBA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBETCA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBETIA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBETOA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBETPA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBETQA
DFHPPT TYPE=ENTRY,PROGRAM=IBMBETTA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFEFCA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFESMA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFESNA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFKCSA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFKMRA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFKPTA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFKTBA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFKTCA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFKTRA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFPGDA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFPMRA
DFHPPT TYPE=ENTRY,PROGRAM=IBMFSTVA
```

Adding DCT Entries

DCT entries are required for queues CPLI and CPLD. CPLI is the queue to which stream output is transmitted, and CPLD is the queue to which PLIDUMPs are sent. These queues may be intrapartition, extrapartition, or indirect. If extrapartition destinations are used, they must be V-format with an LRECL of at least 133 for CPLI and 125 for CPLD.

You use the macro DFHDCT to define the entries for CPLI and CPLD and to define the PLIMSG buffer, which receives the output from the two queues, as shown in the following example:

```
PLIMSG   DFHDCT TYPE=SDSCI,        PL/I OPTIMIZER MESSAGES          X
                BLKSIZE=137,                                        X
                BUFNO=1,                                            X
                DSCNAME=PLIMSG,                                     X
                RECSIZE=133,                                        X
                RECFORM=VARUNB,                                     X
                TYPEFLE=OUTPUT

CPLI     DFHDCT TYPE=EXTRA,        PL/I SYSPRINT OUTPUT             X
                DESTID=CPLI,                                        X
                DSCNAME=PLIMSG

CPLD     DFHDCT TYPE=INDIRECT,     PL/I DUMPS                       X
                DESTID=CPLD,                                        X
                INDDEST=CPLI
```

When DFHDCT encounters the above entry names, it will generate warning messages that state that queues beginning with 'C' are reserved for CICS. These messages are normal; you can ignore them.

Adding PL/I CICS Data Sets to the CICS startup job stream

Before you can run any CICS transactions, you must start CICS in the CICS region. *CICS/OS/VS Version 1 Release 6 Modification 1: Installation and Operations Guide* describes the CICS system startup procedure and gives an example of a CICS startup job stream.

In order to add PL/I's CICS component to CICS, you must concatenate PL/I's CICS data sets after the CICS161.LOADLIB2 and CICS161.LOADLIB data sets in the DFHRPL DD statement in the startup job stream. There are several things you must be aware of when you do this:

1. The PL/I data sets in this concatenation *must* have the same name you assigned to the PL/I transient library CICS components you just installed (for example, PLI.PLILINK).

2. No data set concatenated before PLI.PLILINK (or the transient library CICS component name you chose) may contain any module IBMxxxxx, DFHSAP, or DFHPL1OI.

3. If you have chosen to place your transient library CICS component into SYS1.LINKLIB instead of PLI.PLILINK, no modules named in item 2 above may occur in *any* data set in the DFHRPL concatenation.

After you have added your PL/I CICS data sets to the CICS startup job stream, your PL/I support for CICS is ready to use.

Chapter 8. Changing Option Defaults and Customizing PL/I

After you have ACCEPTed the OS PL/I product(s) you have chosen to install, there are some additional steps you can take to adapt PL/I to your site's specific needs. You can change the IBM-supplied execution-time and/or compiler option defaults, and you can modify the IBM-supplied error exit routine.

Execution-time option defaults are those values assumed if the application programmer does not supply overriding values in JCL at the time of execution of a PL/I program. Compiler option defaults are those values assumed if the application programmer does not supply overriding values in JCL and/or in *PROCESS statements at the time of compilation of a PL/I program.

The error exit routine IBMBEER, as supplied by IBM, does not cause an abend to occur. You can modify the routine so that an abnormal termination of a PL/I program causes an abend.

This chapter contains the information you need to change the execution-time and compiler option defaults and to modify the IBMBEER module. It includes the following information:

- A table listing the execution-time options and the default values assigned to them by IBM.
- Descriptions of the execution-time options and all their possible values.
- Discussion of how to code your changes to the execution-time options macro PLTRLIB in source module IBMBXOPT.
- A table listing the compiler options for both batch and conversational modes, and the default values assigned to them by IBM.
- Descriptions of the compiler options and all their possible values.
- Discussion of how to code your changes to the compiler options macros PLIXOPB and PLIXOPC in source modules IEL0AS and IEL0AT, respectively.
- Discussion of how the IBMBEER module works and an example of how you might modify it.

Changing Execution-Time Option Defaults

The execution-time options and the IBM-supplied defaults provided on the transient library or composite distribution tape are shown in Figure 16. If these values satisfy the needs of your system, you do not need to do anything more.

Option	IBM-Supplied Defaults	
	For Nontasking Environment	For Tasking Environment
HEAP		
Initial allocation	0	0
Incremental value	4K	4K
Location	ANYWHERE	ANYWHERE
HEAP area disposition	KEEP	KEEP
ISAINC (ISA increment)	0	0
Subtask ISA increment	N/A	0
ISASIZE	0	8192
Subtask ISA size	N/A	8192
Subtask maximum number	N/A	20
REPORT/NOREPORT	NOREPORT	NOREPORT
SPIE/NOSPIE	SPIE	SPIE
STAE/NOSTAE	STAE	STAE
TASKHEAP		
Initial allocation	N/A	0
Incremental value	N/A	4K
Location	N/A	ANYWHERE
TASKHEAP area disposition	N/A	KEEP

Figure 16. IBM-Supplied Transient Library Execution-Time Option Default Values

If, however, you prefer different default values, you can change the execution-time option defaults immediately after installing and ACCEPTing the product, or you can run with the IBM-supplied option defaults for a while to see if they meet the needs of your site, and change the defaults at some future time. For more detailed information on these options, see "Description of PLTRLIB Operands" on page 57 and *OS PL/I Optimizing Compiler: Programmer's Guide.*

Note:

You change the execution-time option defaults by RECEIVEing and APPLYing an SMP USERMOD. You can install the USERMOD after you have either APPLYed or ACCEPTed the PL/I product, but we recommend that you wait until after you have ACCEPTed the product, for the following reason: *if you have not ACCEPTed the product, you will not be able to remove the USERMOD using the SMP RESTORE command.*

Specifying Execution-Time Option Defaults

To change the IBM-supplied execution-time option defaults, you must create an SMP USERMOD to modify the PLTRLIB macro statement in the transient library source module IBMBXOPT. This USERMOD uses the IEBUPDTE utility program to replace numbered statements in the IBMBXOPT source module with equal numbered statements from your USERMOD, so you must be sure to *match exactly* the sequence numbers (columns 73 through 80) in your USERMOD to those of the statements in the IBMBXOPT source module they are to replace.

The PLI.INSTALL data set on your distribution tape includes a member that contains a sample USERMOD designed to change the HEAP execution-time option default values for both nontasking and multitasking environments. If you installed the transient library distribution tape, the sample USERMOD is in PLI.INSTALL's member L5MOD. If you installed the composite tape, the sample USERMOD is in member P3MOD.

The following sections describe how to adapt L5MOD or P3MOD to your site's needs.

Description of PLTRLIB Operands

The operands of the PLTRLIB macro statement in the IBMBXOPT ASSEMBLE file define the defaults for the execution-time options. Figure 17 on page 58 gives an example of how the PLTRLIB macro statement is coded.

IBMBXOPT

Macro Operands (IBM-supplied defaults) Sequence

```
PLTRLIB NONTASK=(ISASIZE=0,        BEGIN NONTASK PART  DEFAULT   X10000000
        NOREPORT,                                      DEFAULT   X16000000
        SPIE,                                          DEFAULT   X22000000
        STAE,                                          DEFAULT   X29000000
        HEAP=(0,4K,ANYWHERE,KEEP),                     DEFAULT   X36000000
        ISAINC=(0)),               END OF NONTASK LIST DEFAULT   X43000000
        TASK=(ISASIZE=(8192,8192,20),                  DEFAULT   X50000000
        NOREPORT,                                      DEFAULT   X57000000
        SPIE,                                          DEFAULT   X64000000
        STAE,                                          DEFAULT   X71000000
        HEAP=(0,4K,ANYWHERE,KEEP),                     DEFAULT   X78000000
        TASKHEAP=(0,4K,ANYWHERE,KEEP),                 DEFAULT   X85000000
        ISAINC=(0,0)),                                 DEFAULT   X92000000
        ARCH=XA                                        DEFAULT    95000000
END                                                               99999999
```

Notes:

1. *The actual macro coded in the IBMBXOPT ASSEMBLE source module on your distribution tape might have changed since this book was written, so you must look at that module for sequence numbers before you attempt to make changes to the macro with your USERMOD.*

2. *If you want to change option defaults, you should be aware that ARCH=XA will work on any system, XA or not, so long as you are using Assembler H, Version 2 or later. (See page 64 for more information on the ARCH option.) If you are not using Assembler H, Version 2 or later, you must specify ARCH=STD in your USERMOD, or you will get an assembler error.*

Figure 17. Example of PLTRLIB Macro in IBMBXOPT Source Module

Coding Conventions

The coding conventions used in the example in Figure 17 and in the PLTRLIB macro on your distribution tape are as follows:

- Operands are separated by commas.
- Uppercase letters, numbers, and punctuation marks must be coded exactly as shown.
- Brackets [] must never be coded.
- Lowercase letters and words represent variables for which you must substitute specific information.
- Items or groups of items within brackets [] are optional. They may be omitted at your discretion.

If you are satisfied with the IBM-supplied defaults, leave the macro as it is. If, however, you want to change an option's default value, you must specify its corresponding PLTRLIB operand in your USERMOD. "Coding Your Changes to the PLTRLIB Macro" on page 65 explains how to code your USERMOD to change those values you need to change.

Operand Descriptions

The following section describes the operands of the PLTRLIB macro. Each operand except the last, ARCH, corresponds to an execution-time option. The list will help you determine which option values are most useful for your site's needs and whether you must change some of the IBM-supplied defaults to meet those needs.

NONTASK=([HEAP=(*size*,*increment*,<u>ANYWHERE</u> | BELOW,<u>KEEP</u> | FREE)]
[,ISAINC=*size*]
[,ISASIZE=*size*]
[,REPORT | <u>NOREPORT</u>]
[,<u>SPIE</u> | NOSPIE]
[,<u>STAE</u> | NOSTAE])
specifies the options for tasks that are not multitasking.

HEAP=(*size*,*increment*,<u>ANYWHERE</u> | BELOW,<u>KEEP</u> | FREE)
isolates storage for allocated (that is, CONTROLLED and dynamically allocated BASED) variables from all other PL/I storage and specifies how that storage is to be handled.

size
optional. If specified, it determines the minimum initial size of HEAP storage, and is specified in bytes or as nnnK or as nnM. If not specified, no HEAP area is used. The IBM-supplied default is HEAP=0, that is, the HEAP option is *not* in effect.

increment
optional. If specified, it determines the minimum size of any subsequent increment to the HEAP area. The IBM-supplied default value for the HEAP increment is 4K.

ANYWHERE
specifies that PL/I can allocate the HEAP area anywhere in storage. In an MVS/XA environment, this allows PL/I to allocate HEAP storage either above or below 16 megabytes; PL/I will usually place it above 16 megabytes. In a non-XA environment, use of ANYWHERE necessarily places HEAP storage below 16 megabytes. This is the IBM-supplied default.

BELOW
specifies that PL/I must allocate HEAP storage below 16 megabytes, in storage accessible to 24-bit addressing.

KEEP
specifies that storage allocated to a HEAP increment will *not* be released when a FREE statement in the program deallocates the last variable stored there. This is the IBM-supplied default.

FREE

specifies that storage allocated to a HEAP increment will be released when the variable occupying it is FREEd.

ISAINC=*size*

specifies the minimum size of an increment to the ISA.

If ISAINC is not specified, when the storage currently allocated to the ISA is not large enough to handle all of a program's storage requests, only that amount of storage needed at the time of the request is obtained. When ISAINC is used, the amount of storage allocated when the ISA is too small for the current request is the *larger* of the ISAINC size or the requested size. Thus the use of the ISAINC option can save the increased execution time caused by frequent GETMAINs of small amounts of storage.

size

specifies the minimum amount by which the ISA will be incremented, and is specified in bytes or as nnnK or nnM. The IBM-supplied default is ISAINC=0.

ISASIZE=*size*

specifies the length of the initial storage area.

size

specified in bytes or as nnnK or nnM. It can be preceded by a minus sign. The storage will be contiguous.

A size of '0' causes PL/I to issue a GETMAIN request for the largest block of contiguous storage in the region; PL/I then returns half of that block to the system and retains the other half as its ISA.

The minus sign is used when stating the amount of storage in the partition that must be left outside the resident load module and the ISA. A value of '-0' should not be specified unless the largest possible ISA is required and no files, including SYSPRINT, will be used. Otherwise, an ABEND might occur because of lack of system storage.

For CICS: This is the size of the initial work space obtained. If 0 or -0 is specified, *either by the programmer or because it is the default*, a size is calculated, and a GETMAIN is issued, at execution time. This size is always the PMA (program management area) size plus the main procedure's DSA (dynamic storage area) size.

The IBM-supplied default for ISASIZE in a nontasking environment is ISASIZE=0.

REPORT

specifies that a program management report is required.

NOREPORT
specifies that no program management report is required. This is the IBM-supplied default.

SPIE
specifies that, if a program interrupt occurs, the PL/I error handler is to be used. Under certain circumstances the ERROR condition will be raised. This is the IBM-supplied default.

For CICS: This subparameter is ignored.

NOSPIE
specifies that, on program initialization, PL/I will not issue the SPIE or ESPIE macro to request control after a program check. Unless running under MVS/XA, do not use NOSPIE when extended precision variables are used in the PL/I source program.

For CICS: This subparameter is ignored.

STAE
specifies that, if an abend occurs, the PL/I library routines are to attempt to raise the ERROR condition or to produce a diagnostic message and a PLIDUMP. This is the IBM-supplied default.

For CICS: This specifies that transaction abends will be handled by the PL/I error handler. EXEC CICS HANDLE ABEND command will be issued by PL/I.

NOSTAE
specifies that, on program initialization, PL/I will not issue the STAE or ESTAE macro to request control after an abend.

For CICS: This specifies that transaction abends will not be handled by the PL/I error handler.

TASK=([HEAP=(*size,increment*,<u>ANYWHERE</u> | BELOW,<u>KEEP</u> | FREE)]
[,ISAINC=(*size1,size2*)]
[,ISASIZE=(*size1,size2,tasks*)]
[,REPORT | <u>NOREPORT</u>]
[,<u>SPIE</u> | NOSPIE]
[,<u>STAE</u> | NOSTAE]
[,TASKHEAP=(*size,increment*,<u>ANYWHERE</u> | BELOW,<u>KEEP</u> | FREE)])
specifies the options for use in a multitasking environment.

HEAP=(*size,increment*,<u>ANYWHERE</u> | BELOW,<u>KEEP</u> | FREE)
isolates storage for allocated (that is, CONTROLLED and dynamically allocated BASED) variables from all other PL/I storage and specifies how that storage is to be handled. In a multitasking environment, HEAP option values apply to the major task, only; subtask allocated storage is governed by the TASKHEAP option.

size
is optional. If specified, it determines the minimum initial size of HEAP storage, and is specified in bytes or as nnnK or as nnM. If not specified, no HEAP area is used. The

IBM-supplied default is HEAP=0, that is, the HEAP option is *not* in effect.

increment
: is optional. If specified, it determines the minimum size of any subsequent increment to the HEAP area. The IBM-supplied default value for the HEAP increment is 4K.

ANYWHERE
: specifies that PL/I can allocate the HEAP area anywhere in storage. In an MVS/XA environment, this allows PL/I to locate HEAP storage either above or below 16 megabytes; PL/I will usually place it above 16 megabytes. In a non-XA environment, use of ANYWHERE necessarily places HEAP storage below 16 megabytes. This is the IBM-supplied default.

BELOW
: specifies that PL/I must allocate HEAP storage below 16 megabytes, in storage accessible to 24-bit addressing.

KEEP
: specifies that storage allocated to a HEAP increment will *not* be released when a FREE statement in the program deallocates the last variable stored there. This is the IBM-supplied default.

FREE
: specifies that storage allocated to a HEAP increment will be released when the variable occupying it is FREEd.

ISAINC=(*size1,size2*)
: specifies the minimum size of an increment to the ISA.

If ISAINC is not specified, when the storage currently allocated to the ISA is not large enough to handle all of a program's storage requests, only that amount of storage needed at the time of the request is obtained. When ISAINC is used, the amount of storage allocated when the ISA is too small for the current request is the *larger* of the ISAINC size or the requested size. Thus the use of the ISAINC option can save the increased execution time caused by frequent GETMAINs of small amounts of storage.

size1
: specifies the minimum amount by which the ISA for the major task will be incremented, and is specified in bytes or as nnnK or nnM. The IBM-supplied default is ISAINC=0.

size2
: specifies the minimum amount by which the ISA for any subtask will be incremented, and is specified in bytes or as nnnK or nnM.

ISASIZE=(*size1,size2,tasks*)
: specifies the storage sizes and number of subtasks.

size1

specifies the length of the initial storage area.

This specifies the main task size, in bytes or as nnnK or as nnM. It can be preceded by a minus sign. The storage will be contiguous.

A size of '0' causes PL/I to issue a GETMAIN request for the largest block of contiguous storage in the region; PL/I then returns half of that block to the system and retains the other half as its ISA.

The minus sign is used when stating the amount of storage in the partition that must be left outside the resident load module and the ISA. This storage will be contiguous. A value of '-0' should not be specified unless the largest possible ISA is required and no files, including SYSPRINT, will be used. Otherwise, an ABEND might occur because of lack of system storage.

The IBM-supplied default is 8192 bytes.

size2

specifies the length of each subtask initial storage area. This is an unsigned integer specified in bytes or as nnnK or nnM.

The IBM-supplied default is 8192 bytes.

tasks

is a decimal integer that is the maximum number of subtasks. The IBM-supplied default is 20.

REPORT

specifies that a program management report is to be generated.

NOREPORT

specifies that no program management report is to be generated. This is the IBM-supplied default.

SPIE

specifies that, if a program interrupt occurs, the PL/I error handler is to be used. Under certain circumstances, the ERROR condition will be raised. This is the IBM-supplied default.

NOSPIE

specifies that, on program initialization, PL/I will not issue the SPIE or ESPIE macro to request control after a program check. Unless running under MVS/XA, do not use NOSPIE when extended precision variables are used in the PL/I source program.

STAE

specifies that, if an abend occurs, the PL/I library routines are to attempt to raise the ERROR condition or to produce a diagnostic message and a PLIDUMP. This is the IBM-supplied default.

NOSTAE

specifies that, on program initialization, PL/I will not issue the STAE or ESTAE macro to request control after an abend.

TASKHEAP=(*size*,*increment*,<u>ANYWHERE</u> | BELOW,<u>KEEP</u> | FREE)

specifies that a separate heap storage area is to be created for each subtask in a multitasking environment. This separates storage for CONTROLLED and dynamically allocated BASED variables in a subtask from all other PL/I storage and specifies how that storage is to be handled.

size

is optional. If specified, it determines the minimum initial size of TASKHEAP storage, and is specified in bytes or as nnnK or as nnM. If not specified, no TASKHEAP area is used. The IBM-supplied default is TASKHEAP=0, that is, the TASKHEAP option is *not* in effect.

increment

is optional. If specified, it determines the minimum size of any subsequent increment to the TASKHEAP areas. The IBM-supplied default value for the TASKHEAP increment is 4K.

ANYWHERE

specifies that PL/I can allocate the TASKHEAP areas anywhere in storage. In an MVS/XA environment, this allows PL/I to locate TASKHEAP storage either above or below 16 megabytes; PL/I will usually place it above 16 megabytes. In a non-XA environment, use of ANYWHERE necessarily places TASKHEAP storage below 16 megabytes. This is the IBM-supplied default.

BELOW

specifies that PL/I must allocate TASKHEAP storage below 16 megabytes, in storage accessible to 24-bit addressing.

KEEP

specifies that storage allocated to a TASKHEAP increment will *not* be released when a FREE statement in the program deallocates a variable stored there. This is the IBM-supplied default.

FREE

specifies that storage allocated to a TASKHEAP increment will be released when the variable occupying it is FREEd.

ARCH=STD | <u>XA</u>

specifies whether the transient library will be used on an MVS or an MVS/XA system.

STD

specifies that the transient library will be used on an MVS (non-XA) system.

XA

specifies that the transient library will be used on an MVS/XA system. This is the IBM-supplied default.

If you do not change any execution-time option defaults with a USERMOD, the ARCH=XA default will cause the PL/I load modules to have the appropriate AMODE and RMODE values for the system on which they are installed, whether it is an XA or a non-XA system. You must, however, make changes to the L5ALSMPn job, which allocates SMP data sets, in order to make use of the ARCH=XA specification. See the comments in the L5ALSMPn job you are using.

If you code a USERMOD to change option defaults, you may leave the ARCH=XA specification as it is if you are using Assembler H Version 2 or later, but you must change ARCH=XA to ARCH=STD if you are not using Assembler H Version 2 or later. ARCH=XA specifies AMODE and RMODE instructions; assemblers prior to Assembler H Version 2 do not recognize AMODE and RMODE.

Note: For more information on AMODE and RMODE, see *MVS/XA Conversion Notebook* and *MVS/XA System Programming Library: 31-Bit Addressing.*

Coding Your Changes to the PLTRLIB Macro

You must write an SMP USERMOD to change the option default values you need to change. The L5MOD job and/or the P3MOD job in your PLI.INSTALL data set contains an example of a USERMOD(L5MODn0) written to change the HEAP option default values.

Note:

The remainder of this discussion refers only to L5MOD, but, if you are using P3MOD, the procedure is the same as for L5MOD unless specifically noted otherwise.

Figure 18 shows the sample update statements from the L5MOD job. Notice that the PLTRLIB macro is not reproduced in its entirety in the USERMOD. Only the lines specifying the HEAP option are coded. All the suboperands for HEAP are included on those lines, even the ones not being changed. Notice also that the sequence numbers (columns 73 through 80) of the macro statement operands *match exactly* those of the lines in the IBMBXOPT member shown in Figure 17 on page 58 which they are to replace.

```
./ CHANGE NAME=IBMBXOPT                                          00120000
             HEAP=(40K,4K,ANYWHERE,KEEP),               USER MOD X36000000
             HEAP=(40K,4K,ANYWHERE,KEEP),               USER MOD X78000000
/*                                                               90140000
```

Figure 18. Sample Update Statements for a USERMOD for Changing Execution-Time Option Defaults

Adapting the L5MOD Job to Your Site's Needs

Make a copy of the L5MOD job in PLI.INSTALL so that you can modify it according to your needs.

Examine the copy of source module IBMBXOPT as it exists in PLI.INSTALL to see the PLTRLIB operands and suboperands and their associated sequence numbers (columns 73 through 80). (A listing of IBMBXOPT is included in the LISTJCL output you created earlier.)

Note:

It is *very important* that you look at the IBMBXOPT module in the PLI.INSTALL data set you created from your distribution tape to find the sequence numbers of the macro operands you want to change. Do *not* use sequence numbers from Figure 17 on page 58, as it is possible that the module has been changed since the example in Figure 17 was printed.

The USERMOD uses the IEBUPDTE utility program to update the source of the PLTRLIB macro as coded in the IBMBXOPT module. The update statements you create must conform to the syntax requirements of both IEBUPDTE and assembler source code as well as the rules listed in "Coding Conventions" on page 58. Refer to *MVS/370 Utilities* for information on IEBUPDTE and to the Assembler Language manuals for your system for the rules for coding Assembler Language macros.

Modifying L5MOD: Modify the USERMOD(L5MODn0) to meet your site's requirements. Replace the sample assembler language statements for changing HEAP option default values with your site's execution-time options default update statements. These statements follow the ./ CHANGE NAME=IBMBXOPT control statement.

Selecting SMP Control Statements in L5MOD: If you are using L5MOD, continue with the instructions in this section. If you are using P3MOD, skip to "Selecting SMP Control Statements in P3MOD" on page 67.

You can, if you want to, use the EXEC PROC statement in L5MOD to override L5PROC's symbolic parameters.

L5MOD contains two DD statements that represent the SMP control statements for SMP and SMP/E. These are identified by the comments accompanying them. Find the grouping of input data that matches your site's release level of SMP. Change the DDNAME on the single matching //SMPCNTLn DD * statement to SMPCNTL.

Now go to "Executing Your USERMOD" on page 67.

Selecting SMP Control Statements in P3MOD: P3MOD is designed to apply USERMODs for compiler option defaults modules IEL0AS and IEL0AT in addition to the execution-time option defaults module IBMBXOPT. If you are using P3MOD to change compiler option defaults, skip ahead now to "Changing Compiler Option Defaults." If you are using P3MOD to change *only* execution-time option defaults, you must remember to delete the USERMODs for compiler option defaults. These are the ++ USERMOD(P1MODn0) and the ++ USERMOD(P1MODn4) statements and the statements following and associated with them.

You can, if you want to, use the EXEC PROC statement in P3MOD to override P3PROC's symbolic parameters.

P3MOD contains several DD statements that represent the SMP control statements for all the possible combinations of SMP level and USERMODs being used. These are identified by the comments accompanying them. Find the grouping of input data that matches your site's release level of SMP and the USERMOD you are using. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL. The other //SMPnnn DD statements will have no effect.

Executing Your USERMOD: Now execute job L5MOD or P3MOD to RECEIVE and APPLY your modifications to IBMBXOPT.

If you have violated any of the rules for coding your changes to PLTRLIB, an appropriate MNOTE will be issued when IBMBXOPT is reassembled during the APPLY step. Correct and recode any statement in the L5MOD or P3MOD job that contains an error. Execute an SMP or SMP/E REJECT to restore the PLTRLIB macro to its original state (see *OS/VS SMP Programmer's Guide* or *SMP/E Reference* for more information on REJECT). Then reexecute the job to RECEIVE and APPLY your USERMOD.

Note:

Do *not* ACCEPT your USERMOD. This would overwrite the original data sets distributed by IBM. After you have APPLYed the USERMOD, it is ready for use.

Changing Compiler Option Defaults

This section describes the compiler options for the OS PL/I Optimizing Compiler and discusses how to change their IBM-supplied defaults.

The compiler options and their default values as supplied by IBM on the distribution tape are shown in Figure 19 on page 68. The options shown in the figure are coded as operands of the two macros that are used to set their default values. Batch mode option defaults are set by macro PLIXOPB, and conversational mode (TSO) option defaults are set by macro PLIXOPC. These macros are discussed further under "Specifying Compiler Option Defaults" on page 69 and "Coding Your Changes to the PLIXOPB and PLIXOPC Macros" on page 75.

Compiler Option	IBM-Supplied Defaults Batch Mode	TSO
*PRINT={NO|TERM|SYS}|([{NO|TERM| SYS}][,[class][,[n][,m]]])	Not applicable	(NO,A,300, 100)
AGGREGA={YES|NO}	NO	NO
ATTRIBU={YES|NO}	NO	NO
CHARSET=([{EBCDIC|BCD}][,{48|60}])	(EBCDIC,60)	(EBCDIC,60
COMPILE={YES|NO|NOW|NOE|NOS}	NOS	NOS
CONTROL='charstring'	OPTIMIZE	OPTIMIZE
COUNT={YES|NO}	NO	NO
DECK={YES|NO}	NO	NO
ESD={YES|NO}	NO	NO
FLAG={I|W|E|S}	I	W
FLOW=({YES|NO}[,n,p])	(NO,25,10)	(NO,25,10)
*FMARGIN=({m,n}[,c])	(2,72)	(2,72)
*FSEQUEN={(m,n)|NO}	(73,80)	(73,80)
GONUMBE={YES|NO}	NO	NO
GOSTMT={YES|NO}	NO	NO
*GRAPHIC=(pad,quote,g,ld,rd)	(4040,427D, 42C7,0E,0F)	(4040,427D, 42C7,0E,0F)
IMPRECI={YES|NO}	NO	NO
INCLUDE={YES|NO}	NO	NO
INSOURC={YES|NO}	YES	NO
INTERRU={YES|NO}	NO	NO
LINECOU=n	55	55
LIST={YES|NO}	NO	NO
LMESSAG={YES|NO}	YES	NO
MACRO={YES|NO}	NO	NO
MAP={YES|NO}	NO	NO
MARGINI={NO|'character'}	NO	NO
MDECK={YES|NO}	NO	NO
NEST={YES|NO}	NO	NO
NUMBER={YES|NO}	NO	YES
OBJECT={YES|NO}	YES	YES
OFFSET={YES|NO}	NO	NO
OPTIMIZ=[{NO|TIME}]|[{0|2}]	NO|0	NO|0
OPTIONS={YES|NO}	YES	NO
SIZE={n|nk|MAX}	MAX	MAX
SOURCE={YES|NO}	YES	NO
STMT={YES|NO}	YES	NO
STORAGE={YES|NO}	NO	NO
SYNTAX={YES|NO|NOW|NOE|NOS}	NOS	NOS
TERMINA={YES|NO|(option list**)}	NO	YES
*TSTAMP={YES|NO}	NO	NO
*VMARGIN=({m,n}[,c])	(10,100)	(10,100)
*VSEQUEN={(m,n)|NO}	(1,8)	(1,8)
XREF={YES|NO}|({YES|NO}[,...]])	(NO,F)	(NO,F)
*DELETE=(item[,item[,...])	-	-

* Options used at installation time only. These options are not available to the application programmer.

** See *OS Pl/I Optimizing Compiler: Programmer's Guide* for a list of valid options.

Figure 19. Compiler Options and Their IBM-Supplied Defaults

If the IBM-supplied compiler option default values satisfy the needs of your system, you do not need to do anything further. If, however, you prefer different default values, you can change the compiler option defaults now, or you can run with the IBM-supplied option defaults for a while to see if they meet the needs of your site, and change the default values at some future time. For more detailed information on the compiler options used for application programming, see *OS PL/I Optimizing Compiler: Programmer's Guide*. For more information on the options used only at time of installation, see "Operands Representing Compiler Options Whose Use is Restricted" on page 72.

Note:

You change the compiler option defaults by RECEIVEing and APPLYing SMP USERMODs for batch and/or conversational mode option defaults. You can install the USERMOD(s) after you have either APPLYed or ACCEPTed the PL/I product, but we recommend that you wait until after you have ACCEPTed the product, for the following reason: *if you have **not** ACCEPTed the product, you will not be able to remove the USERMOD(s) using the SMP RESTORE command.*

Allowing for Compiler Option Interdependencies

Some compiler options, such as STMT, GOSTMT, and NUMBER, are interdependent at compile-time. If, for example, you specify STMT at compile-time to override the IBM-supplied default of NOSTMT (STMT=NO), the compiler automatically performs the compilation as if you had also specified GOSTMT and NONUMBER, even if you did not. These interdependencies are described fully in *OS PL/I Optimizing Compiler: Programmer's Guide.*

The automatic adjustment of one compiler option based on the setting of another compiler option does *not* take place at installation time. If you code your USERMOD to change the IBM-supplied default value for STMT from STMT=NO to STMT=YES, you must also change the defaults for NUMBER and GOSTMT.

Specifying Compiler Option Defaults

To change the IBM-supplied compiler option defaults, you must create SMP USERMODs to modify the PLIXOPB and PLIXOPC macro instructions in the compiler source modules IEL0AS and IEL0AT. Source module IEL0AS contains the PLIXOPB macro, whose operands specify the compiler option defaults that will be in effect in batch mode. Source module IEL0AT contains the PLIXOPC macro, whose operands specify the compiler option defaults that will be in effect in conversational mode, that is, under TSO.

The SMP USERMODs use the IEBUPDTE utility program to replace numbered statements in the IEL0AS and IEL0AT modules with equal numbered statements from your USERMODs, so you must be sure to match exactly the sequence numbers (columns 73 through 80) in your USERMODs to those of the statements they are to replace in the IEL0AS and IEL0AT source modules.

The PLI.INSTALL data set on your distribution tape includes a job that contains a sample USERMOD(P1MODn0), designed to change some batch mode compiler option defaults, and a sample USERMOD(P1MODn4), designed to change a conversational mode compiler option default. If you installed the individual compiler distribution tape, the sample USERMODs are in the P1MOD job in PLI.INSTALL. If you installed the composite tape, the sample USERMODs are in job P3MOD, along with USERMOD(L5MODn0) for changing execution-time option defaults. The following sections describe how to adapt P1MOD or P3MOD, to your site's needs.

Description of PLIXOPB and PLIXOPC Operands

The operands of the PLIXOPB and PLIXOPC macro statements define the defaults for the compiler options. Figure 20 on page 71 gives an example of how the PLIXOPB macro statement is coded.

IEL0AS

```
Macro    Operand (IBM-supplied default)                                     Sequence

PLIXOPB AGGREGA=NO,                                      DEFAULT      X00010000
        ATTRIBU=NO,                                      DEFAULT      X00020000
        CHARSET=(EBCDIC,60),                             DEFAULT      X00030000
        COMPILE=NOS,                                     DEFAULT      X00040000
        CONTROL='OPTIMIZE',                              DEFAULT      X00050000
        COUNT=NO,                                        DEFAULT      X00060000
        DECK=NO,                                         DEFAULT      X00070000
        ESD=NO,                                          DEFAULT      X00080000
        FLAG=I,                                          DEFAULT      X00090000
        FLOW=(NO,25,10),                                 DEFAULT      X00100000
        FMARGIN=(2,72),                                  DEFAULT      X00110000
        FSEQUEN=(73,80),                                 DEFAULT      X00120000
        GONUMBE=NO,                                      DEFAULT      X00130000
        GOSTMT=NO,                                       DEFAULT      X00140000
        GRAPHIC=(4040,427D,42C7,0E,0F),                  DEFAULT      X00150000
        IMPRECI=NO,                                      DEFAULT      X00160000
        INCLUDE=NO,                                      DEFAULT      X00170000
        INSOURC=YES,                                     DEFAULT      X00180000
        INTERRU=NO,                                      DEFAULT      X00190000
        LINECOU=55,                                      DEFAULT      X00200000
        LIST=NO,                                         DEFAULT      X00210000
        LMESSAG=YES,                                     DEFAULT      X00220000
        MACRO=NO,                                        DEFAULT      X00230000
        MAP=NO,                                          DEFAULT      X00240000
        MARGINI=NO,                                      DEFAULT      X00250000
        MDECK=NO,                                        DEFAULT      X00260000
        NEST=NO,                                         DEFAULT      X00270000
        NUMBER=NO,                                       DEFAULT      X00280000
        OBJECT=YES,                                      DEFAULT      X00290000
        OFFSET=NO,                                       DEFAULT      X00300000
        OPTIMIZ=NO,                                      DEFAULT      X00310000
        OPTIONS=YES,                                     DEFAULT      X00320000
        SIZE=MAX,                                        DEFAULT      X00330000
        SOURCE=YES,                                      DEFAULT      X00340000
        STMT=YES,                                        DEFAULT      X00350000
        STORAGE=NO,                                      DEFAULT      X00360000
        SYNTAX=NOS,                                      DEFAULT      X00370000
        TERMINA=NO,                                      DEFAULT      X00380000
        TSTAMP=NO,                                       DEFAULT      X00390000
        VMARGIN=(10,100),                                DEFAULT      X00400000
        VSEQUEN=(1,8),                                   DEFAULT      X00410000
        XREF=(NO,F)                                      DEFAULT       00420000
*       DELETE=( , , , ,   )                             DEFAULT       00430000
END                                                                    00440000
```

Notes:

1. *The actual macro coded in the IEL0AS source module on your distribution tape may have changed since this book was written, so you must look at that module before you attempt to make changes with your USERMOD.*

2. *Default values will always be used at compilation time for any options specified in the DELETE option. If you plan to use the DELETE option, you must make some changes to IEL0AS before assembling it. You must remove the asterisk from column one of the DELETE line, add a comma after the XREF specification, and add a continuation character in column 72 of the XREF line.*

Figure 20. Example of PLIXOPB Macro in IEL0AS Source Module

Coding Conventions

The coding conventions used in Figure 19 on page 68, in the example in Figure 20 on page 71, in the PLIXOPB and PLIXOPC macros on your distribution tape, and in the following descriptions, are as follows:

- Operands are separated by commas.
- Uppercase letters, numbers, and punctuation marks must be coded exactly as shown.
- Brackets [], and braces {}, must never be coded.
- Lowercase letters and words represent variables for which you must substitute specific information.
- Groups of items within braces {} are related. One of the items must be coded.
- Items or groups of items within brackets [] are optional. They may be omitted at your discretion.

If you are satisfied with the IBM-supplied defaults, leave the macros as they are. If, however, you want to change an option's default value, for use in batch mode or under TSO, you must specify its corresponding PLIXOPB or PLIXOPC operand in the appropriate USERMOD. "Coding Your Changes to the PLIXOPB and PLIXOPC Macros" on page 75 explains how to code your USERMOD(s) to change those values you need to change.

Operands Representing Compiler Options Whose Use is Restricted

Most of the operands of the PLIXOPB and PLIXOPC macro instructions represent compiler options that are described in *OS PL/I Optimizing Compiler: Programmer's Guide*.

Some compiler options, however, designated by a "*" in Figure 19, can not be used by the application programmer; they are used at installation time, only. These are described below.

PRINT={<u>NO</u> | TERM | SYS} | ([{NO | TERM | SYS}][,[class] [,[n][,m]]])
specifies how SYSPRINT data set is to be allocated when the optimizing compiler is started from TSO.

TERM
specifies that SYSPRINT is to be allocated to the terminal.

<u>NO</u>
specifies that SYSPRINT is to be allocated to a DUMMY data set. This is the IBM-supplied default.

SYS
specifies that SYSPRINT is to be allocated to the SYSOUT class specified by class.

class

specifies the SYSOUT class for SYSPRINT. A is the IBM-supplied default.

n

specifies the primary allocation in number of lines. 300 is the IBM-supplied default.

m

specifies the secondary allocation in number of lines. 100 is the IBM-supplied default.

FMARGIN=(*m*,*n*[,*c*])

specifies that MARGINS(m,n,c) will be the default when the source is read from fixed-length records. The IBM-supplied default for this option is (2,72).

FSEQUEN=NO | (*m*,*n*)

specifies defaults for the SEQUENCE compiler option if the source input file is F-format.

NO

specifies that NOSEQUENCE will be the compiler option default for F-format input records.

(*m*,*n*)

specifies that SEQUENCE (m,n) will be the compiler option default.

The IBM-supplied default for this option is (73,80).

GRAPHIC=(pad,quote,g,ld,rd)

used to replace IBM defaults for the graphic padding, left delimiter, right delimiter, graphic quotation mark, and the graphic "G" for graphic string data. An example of specifying the GRAPHIC operand is:

```
GRAPHIC=(4040,427D,42C7,0E,0F)
```

Note: If you use the shared library, the GRAPHIC options you specify here *must match* those you specify on the PLRSHR macro used to generate the shared library.

pad

specifies the hexadecimal padding. The first byte of the padding must be the same as the second byte. For example, X'5050' is valid but X'5060' is not. The IBM-supplied default is X'4040'.

quote

is the hexadecimal representative of the graphic quotation mark. The IBM-supplied default is X'427D'.

g

is the hexadecimal representation of the graphic "G." The IBM-supplied default is X'42C7'.

Note: If the preprocessor is used, you may not specify values X'00' through X'06' in either byte of the graphic quotation mark ("quote" option) or graphic "G" ("g" option).

ld

specifies the hexadecimal left delimiter. The IBM-supplied default is X'0E'. You must use X'0E' for the left delimiter, since this is an IBM hardware-generated character.

rd

specifies the hexadecimal right delimiter. The IBM-supplied default is X'0F'. You must use X'0F' for the right delimiter, since this is an IBM hardware-generated character.

Note: If the preprocessor is used, you may not specify values X'00' through X'06' in the left or right delimiter ("ld" or "rd" options). The left or right delimiter also cannot be set to any character in the 60-character set described in *OS and DOS PL/I Language Reference Manual.*

TSTAMP=YES | NO

specifies whether or not the compiler is to place the time and date of compilation in the STATIC INTERNAL CSECT, in a location indicated by the first word in the STATIC INTERNAL CSECT. This provides a time-sharing facility if you prefer not to use the COMPILETIME preprocessor function in all your programs.

YES

specifies that the time and date of compilation are to be placed in the STATIC INTERNAL CSECT.

NO

specifies that the time and date of compilation are not to be placed in the STATIC INTERNAL CSECT. This is the IBM-supplied default.

VMARGIN=(*m*,*n*[,*c*])

specifies that MARGINS(m,n,c) will be the compiler option default when the source is read from variable- or undefined-length records.

The IBM-supplied default for this option is (10,100).

VSEQUEN=NO | (*m*,*n*)

specifies defaults for the SEQUENCE compiler options if the first source input file is V- or U-format.

NO

specifies that NOSEQUENCE will be the compiler option default for V- or U-format input records.

(*m*,*n*)

specifies that SEQUENCE (m,n) will be the compiler option default.

The IBM-supplied default for this option is (1,8).

DELETE=(*item*[,*item*[,...]])

specifies that the options in the list cannot be used at compilation time in the *PROCESS statement or PARM field of the EXEC statement to override the default options established by the PLIXOPB and PLIXOPC macro instructions, unless the CONTROL option is also specified with the correct password. One or more of the following options can be specified. All but DUMP, which cannot be used at installation time, are shown in Figure 19 on page 68.

```
AGGREGA    FLOW       INTERRU    NEST       STMT
ATTRIBU    FMARGIN    LINECOU    NUMBER     STORAGE
CHARSET    FSEQUEN    LIST       OBJECT     SYNTAX
COMPILE    GONUMBE    LMESSAG    OFFSET     TERMINA
COUNT      GOSTMT     MACRO      OPTIMIZ    VMARGIN
DECK       GRAPHIC    MAP        OPTIONS    VSEQUEN
DUMP       IMPRECI    MARGINI    SIZE       XREF
ESD        INCLUDE    MDECK      SOURCE
FLAG       INSOURCE
```

Coding Your Changes to the PLIXOPB and PLIXOPC Macros

You must write an SMP USERMOD to change the batch mode option default values you need to change, and a separate USERMOD to change the conversational mode (TSO) option default values you need to change. The P1MOD job and/or the P3MOD job in your PLI.INSTALL data set contains examples of USERMODs written to change option default values for both batch mode and TSO compilations.

Note:

The remainder of this discussion refers only to P1MOD, but, if you are using P3MOD, the procedure is the same as for P1MOD unless specifically noted otherwise.

Figure 21 shows sample update statements for batch mode compiler option defaults. Notice that the PLIXOPB macro is not reproduced in its entirety in the USERMOD. Only the lines that specify the option defaults to be changed are written. Notice also that the sequence numbers (columns 73 through 80) of the macro statement operands *match exactly* those of the lines they are to replace in the IEL0AS member shown in Figure 20.

```
./ CHANGE NAME=IEL0AS                                       00001200
|               MARGINI='|',                        USERMOD X00250000
                NEST=YES,                           USERMOD X00270000
                TSTAMP=YES,                         USERMOD X00390000
/*
```

Figure 21. Sample USERMOD for Changing Batch Mode Compiler Option Defaults

Adapting the P1MOD Job to Your Needs

Make a copy of the P1MOD job in PLI.INSTALL so that you can modify it according to your needs.

If your site does not use TSO, you will be interested only in the information about IEL0AS, the batch mode macro source module. Delete from your copy of P1MOD the USERMOD(P1MODn4) statement and all the lines following it that pertain to IEL0AT.

If your site uses TSO, you must determine the compiler options your site needs for both batch and TSO processing.

The following material discusses the procedure for coding the USERMOD(P1MODn0) job for batch mode options, only. The procedure for coding the USERMOD(P1MODn4) job for TSO options is exactly the same, except that the module name is IEL0AT.

Matching Sequence Numbers: Examine the copy of source module IEL0AS as it exists in PLI.INSTALL to see the PLIXOPB operands and suboperands and their associated sequence numbers (columns 73 through 80). (A listing of IEL0AS is included in the LISTJCL output you created earlier.)

Note:

It is *very important* that you look at the IEL0AS module in the PLI.INSTALL data set you created from your distribution tape to find the sequence numbers of the macro operands you want to change. Do *not* use sequence numbers from Figure 20 on page 71, as it is possible that the module has been changed since the example in that figure was printed.

The USERMOD uses the IEBUPDTE utility program to update the source of the PLIXOPB macro as coded in the IEL0AS module. The update statements you create must conform to the syntax requirements of both IEBUPDTE and assembler source code as well as the rules listed in "Coding Conventions" on page 72. Refer to *MVS/370 Utilities* for information on IEBUPDTE and to the Assembler Language manuals for your system for the rules for coding Assembler Language macros.

Modifying P1MOD: Modify the P1MOD job to meet your site's requirements. Replace the sample assembler language statement for changing an option's default value (as shown in Figure 21 on page 75) with statements specifying your site's desired compiler option defaults. These statements follow the ./ CHANGE NAME=IEL0AS control statement.

If you are also changing TSO option defaults, put your replacement statements for those after the ./ CHANGE NAME=IEL0AT control statement.

If your site does not use TSO, or if you are not changing TSO option defaults, you must remember to delete the statements relating to IEL0AT. In the same way, if you are changing TSO option defaults but not those for batch mode, you must delete those statements relating to IEL0AS.

Selecting SMP Control Statement in P1MOD: If you are using P1MOD, continue with the instructions in this section. If you are using P3MOD, skip to "Selecting SMP Control Statements in P3MOD" on page 77.

You can, if you want to, use the EXEC PROC statement in P1MOD to override P1PROC's symbolic parameters.

P1MOD contains several DD statements that represent the SMP control statements for all the possible combinations of SMP level and USERMODS being used. These are identified by the comments accompanying them. Find the group of input data that matches your site's release level of SMP and the set of options (batch mode and/or TSO) you are using. Change the DDNAME on the single matching //SMPCNTLn DD * statement to SMPCNTL.

Now go to "Executing Your USERMOD(s)."

Selecting SMP Control Statements in P3MOD: P3MOD is designed to apply USERMODs for execution-time option defaults module IBMBXOPT in addition to the compiler option defaults modules IEL0AS and IEL0AT. If you are using P3MOD to change option defaults in one or two, but not all three, modules, you must remember to delete the USERMOD statements and the lines following them that relate to the unchanged modules, before you run P3MOD.

You can, if you want to, use the EXEC PROC statement in P3MOD to override P3PROC's symbolic parameters.

P3MOD contains several DD statements that represent the SMP control statements for all the possible combinations of SMP level and USERMODs being used. These are identified by the comments accompanying them. Find the grouping of input data that matches your site's release level of SMP and the USERMODs you are using. Change the DDNAME on the single matching //SMPnnn DD * statement to SMPCNTL. The other //SMPnnn DD statements will have no effect.

Executing Your USERMOD(s): Now execute job P1MOD or P3MOD to RECEIVE and APPLY your modifications to IEL0AS and/or IEL0AT.

If you have violated any of the rules for coding your changes to PLIXOPB or PLIXOPC, an appropriate MNOTE will be issued when IEL0AS or IEL0AT is reassembled during the APPLY step. Remember that some of the compiler options are interdependent, as explained in "Allowing for Compiler Option Interdependencies" on page 69. If you coded your USERMOD to change the IBM-supplied default value for one option without changing those that relate to it, an IELnnnn error message will explain the inconsistency in the option settings. You must change the noted option defaults accordingly. Correct and recode any statement in the P1MOD or P3MOD job that contains an error. Execute an SMP or SMP/E REJECT to restore the PLIXOPB and/or PLIXOPC macros to their original state (see *OS/VS SMP Programmer's Guide* or *SMP/E Reference* for more information on REJECT). Then reexecute the P1MOD job to RECEIVE and APPLY your USERMOD.

Note:

Do *not* attempt to ACCEPT your USERMOD. This will overwrite the original data sets distributed by IBM. After you have APPLYed the USERMOD, it is ready for use.

Rerunning the Sample Program

After APPLYing your USERMOD(s), you might want to rerun the sample program to see that the new option defaults are in effect. See Chapter 6, "Running the Installation Verification Program" on page 47.

Modifying the PL/I-Supplied Error Exit Routine

IBM supplies as part of the resident library a PL/I error exit routine named IBMBEER. You can use the IBMBEER routine to terminate a PL/I application program with an abnormal termination condition. This exit will be entered when the ERROR condition is raised and the standard system action is taken, that is, under one of the following conditions:

- No ERROR on-unit
- An ERROR on-unit that does not terminate with a GOTO, STOP, or EXIT statement

If you need an abnormal termination for the situation described above, you must modify the IBM-supplied module IBMBEER in the PL/I Resident Library, as indicated below.

Altering the IBM-Supplied IBMBEER Module

IBMBEER is called by the program termination routine under the conditions listed above. On return from IBMBEER, the program termination routine inspects the contents of register 15 and, if not zero, executes an ABEND macro, using the contents of register 15 as the argument of ABEND.

The IBM-supplied IBMBEER module sets register 15 to zero and returns. Thus an abend does not occur. If you want an abend to occur, you must modify the IBMBEER module to set register 15 to a value that can be passed to the ABEND macro as an argument.

The first byte of the value passed to ABEND is a flag byte. The flag byte has the following meanings: The flag byte has the following meanings:

```
1xxx xxxx = DUMP (128)
0xxx xxxx = NO DUMP (0)
x1xx xxxx = STEP ABEND (64)
x0xx xxxx = TASK ABEND (0)
```

The last three bytes of the value passed to ABEND contain a value that will become the ABEND code. The value must be greater than 0 and less than 4096.

Figure 22 on page 79 shows a replacement module that will cause a step abend with a different ABEND code depending on whether an ERROR on-unit has been executed. Note that the member name of IBMBEER in the PL/I Resident Library is IBMBEERA.

```
IBMBEER1  CSECT                     CSECT NAME
IBMBEER1  AMODE  ANY
IBMBEER1  RMODE  ANY
          ENTRY  IBMBEERA           DEFINE ENTRY POINT
          USING  *,15               ADDRESSABILITY
IBMBEERA  EQU    *                   ENTRY POINT
          LTR    1,1                 BRANCH
          BM     NOERR
          L      15,RETNCOD1        ERROR ON-UNIT
          BR     14
NOERR     EQU    *
          L      15,RETNCOD2        NO ERROR ON-UNIT
          BR     14
*
DUMP      EQU    128                128=DUMP, 0=NO DUMP
STEP      EQU    64                 64=STEP ABEND, 0=TASK ABEND
USERCOD1  EQU    3001               USER COMPLETION CODES (MUST BE
USERCOD2  EQU    3002               GREATER THAN 0 AND LESS THAN 4096)
          DS     0F
RETNCOD1  DC     AL1(DUMP+STEP)
          DC     AL3(USERCOD1)
RETNCOD2  DC     AL1(DUMP+STEP)
          DC     AL3(USERCOD2)
          END
```

Notes:

1. *Only Assembler H Versions 2 and later recognize AMODE and RMODE statements. If you are not using this assembler, or you do not need XA support, omit these statements.*

2. *If this module is used to replace the IBM-supplied module IBMBEER in the resident library and if the program is terminated because of the ERROR condition, an abend will occur. The ABEND code will be 3001 if there was an ERROR on-unit and 3002 if there was not.*

Figure 22. Example of a User-Written IBMBEER Replacement Module

You can include your IBMBEER module in the resident library, PLI.PLIBASE, by link-editing with the object module for the user's IBMBEER, as shown in Figure 23, or by using an SMP USERMOD statement.

```
//         EXEC LKED,PARM='LIST,NCAL,RENT,REFR,SIZE=(128K,24K)'
//BEER     DD   DSN=IBMBEER.OBJECT.MODULE,DISP=SHR
//SYSLMOD DD    DSN=PLI.PLIBASE,DISP=OLD
//SYSLIN   DD   *
 INCLUDE BEER
 ENTRY IBMBEERA
 NAME IBMBEERA(R)
/*
```

Figure 23. JCL to Include User-Written IBMBEER in PLI.PLIBASE

Note: SMP does not keep track of a user-written IBMBEER. Therefore, if you apply corrective or preventive service to the resident library, you must repeat the above procedure to include your IBMBEER module.

Tuning Your OS PL/I Product

There are two things you can do after you install the OS PL/I product to improve its performance. These are:

- Using the shared library, and
- Adding transient library modules to the link pack area

You can also disable the PL/I fast initialization facility if storage for PL/I program load modules is limited. These procedures are discussed in Chapter 9, "Postinstallation Tuning" on page 81.

Chapter 9. Postinstallation Tuning

Tuning is the process of adjusting your product, or parts of it, to improve its efficiency and/or the overall performance of your system. This chapter discusses some techniques for tuning the OS PL/I libraries and compiler to best suit the needs of your site. It contains discussions of the following:

- Using the shared library
  - Establishing compatibility between release levels
  - Creating the shared library
  - Loading the shared library into main storage
  - Using IBM-supplied cataloged procedures
  - Creating the shared library for use under CICS
- Adding transient library modules to the link pack area
- Disabling the PL/I fast initialization/termination facility if storage for PL/I program load modules is limited at your site

Using the Shared Library

OS PL/I allows you to select groups of resident library modules and place them in a shared library in the MVS link pack area. During link-edit of a PL/I program's object module, a program stub is substituted for each resident library module referred to that resides in the link pack area. When multiple PL/I application programs execute concurrently, they can share a single copy of any resident library module residing in the link pack area.

Use of the shared library is optional. If you choose to use it, you can create your shared library at any time after you have executed the SMP APPLY command to install the resident library.

Deciding Whether to Create the Shared Library

If your site executes many OS PL/I application programs concurrently, using the shared library can be beneficial. There are also, however, some disadvantages to using the shared library. The advantages and disadvantages of using the shared library are listed below:

Advantages:

- Composite load modules representing OS PL/I application programs need less DASD storage when the PL/I applications use resident library routines that reside in the shared library.
- These smaller composite load modules need less time to be loaded into storage for execution.
- These smaller composite load modules need less main storage.
- If all the resident library modules are in the shared library, corrective or preventive maintenance to resident library modules in the shared library becomes available to PL/I applications after the next initial program load (IPL), without the applications needing to be link-edited again.
- Sites that use primarily PL/I applications might see a decrease in paging rate because frequently used resident library modules are more likely to remain in main storage.

Disadvantages:

- The shared library is not supported by SMP. Therefore, when you apply corrective or preventive maintenance or a USERMOD to a resident library module residing in the shared library, you must rebuild your shared library so that it will contain the updated module. You can rebuild the shared library by rerunning Job 2 as described in "Running Job 2" on page 95.
- If you do not place all the resident library modules in the shared library, it is possible that applying a program temporary fix (PTF) can cause a load module execution to fail because it uses resident library modules at different service levels. This can happen if a PTF affects more than one module, one or more but not all of which are in the shared library.

  If you rebuild the shared library but forget to link-edit again applications that use modules not in the shared library, the resident library modules not at the same service level might no longer be compatible.

Establishing Compatibility Between Releases When Using Shared Library

Different release levels of the optimizing compiler and the libraries will produce compatible executions only if the options specified for the shared library are *exactly* the same as those specified at any previous generation of the shared library, and only if *all* resident library modules were in the old shared library and you are placing *all* of them in the new shared library.

When you install a new release of the resident library, the modules in the link pack area are still from the old release, so programs link-edited at that release level will continue to work. When you create a new shared library, if a previously link-edited program contains *only* program stub pointers to the shared library, then the program will continue to work with the new shared library as if it had been link-edited again with the new library.

If, however, your program load module contains a mixture of program stubs and complete modules from the old resident library, you have a mixture of releases; this can cause failures at execution time.

If not all of the resident library modules are in the shared library, then all programs previously compiled and link-edited using the PLISHRE module from the old shared library must be link-edited with the new shared library.

Compatibility between Shared Libraries Generated with Different Options

In general, if a program was link-edited with a shared library generated with one set of options, it will fail in execution if you run it on a system that has a different set of shared library options specified. You must link-edit the program again on the new system before you attempt to execute it.

Creating the Shared Library

To build the shared library, you must run two jobs. The first job, which you create, assembles a macro instruction in which you have specified the groups of modules you require in the shared library, and the environment (tasking, nontasking, or both) in which you wish to use it. The second job, which is assembler output from Job 1, creates five modules and link-edits four into the SYS1.PLIBASE link library and one into the SYS1.PLITASK link library or to the target libraries you chose when you installed the resident library.

Three of the modules Job 2 places in SYS1.PLIBASE (or your equivalent library) belong in the link pack area. There are two ways to put them there:

- Include the modules in SYS1.LPALIB, or
- Use the MLPA (modified link pack area) facility of MVS.

These methods are described in "Loading the Shared Library into Common Storage" on page 96.

Before you can use the shared library modules in the link pack area, you must perform an initial program load (IPL). For information on using the shared library, see *OS PL/I Optimizing Compiler: Programmer's Guide.*

Note: If you are using the XF Assembler to create your shared library, you will need to restrict the number of modules you put in it. "Using the XF Assembler to Create a Shared Library" on page 97 describes how to calculate the number of modules you can put in your shared library.

Running Job 1

The first step in the creation of the shared library is to select the groups of modules that you require in the shared library (see "Shared Library Macro Operand Descriptions" on page 85 for groups of modules available). You then specify the groups of modules in the PLRSHR macro instruction as shown in Figure 24.

As noted above, if you are using the XF Assembler, you must limit the number of modules you specify. If you do not have an extended architecture system, you might also want to limit the modules you specify to frequently used modules only, to make the best use of the storage available. Note that the housekeeping and basic error handling modules are always included in the shared library.

If you can, however, and especially if you have the storage constraint relief of MVS/XA, we recommend that you place all of your resident library into the shared library. Remember, though, that SMP does not keep track of modules in the link pack area, so when you apply maintenance to resident library modules in the link pack area, you must rerun Job 2 to rebuild the shared library and ensure that those modules are updated.

```
//SHRLIB    JOB   ---
//         EXEC PGM=ASMBLR,PARM=(DECK,NOLOAD)
//SYSUT1     DD UNIT=SYSSQ,DISP=(NEW,PASS),
//              SPACE=(1024,(500,150),,,ROUND),DSN=&&SYSUT1
//SYSUT2     DD UNIT=SYSSQ,DISP=(NEW,PASS),
//              SPACE=(1024,(300,150),,,ROUND),DSN=&&SYSUT2
//SYSUT3     DD UNIT=SYSDA,DISP=(NEW,PASS),
//              SPACE=(1024,(300,150),,,ROUND),DSN=&&SYSUT3
//SYSPRINT   DD SPACE=(121,(500,40),RLSE),SYSOUT=A,
//              DCB=(RECFM=FB,LRECL=121,BLKSIZE=3509)
//SYSPUNCH   DD UNIT=TAPE,LABEL=(1,NL)
//SYSLIB     DD DSN=PLI.SHRMAC,DISP=SHR
//SYSIN      DD *
         ICTL 1,71,3
        PUNCH 'SPECIFY HERE JOB STATEMENT OF YOUR OWN CHOOSING'
       PLRSHR MODES=(BASE),ARRAY=(BASIC.LOGIC.LEAF,                   X
  FIXED.PROD.SUM,FLOAT.PROD.SUM.POLY,EXTND.PROD.SUM.POLY),            X
  CONV=(ARITH.CHAR.DEC.BIN.BIT,EDIT,BIT.CHAR.BIN,                     X
  CHAR.ARITH.BIT.PIC,PIC.CHECK.DEC.BIT,EXTND,COMPLEX),                X
  IO=(DATA.OUTPUT,EDIT.OUTPUT),                                       X
  JOBSTMT=SUPPLIED,                                                   X
  MFUNC=(RETC,TIME,DATE,DELAY),                                       X
  STRGS=(BIT.LOGIC.COMPARE.ASSIGN.INDEX.SUBSTR,                       X
  CHAR.INDEX.TRANS.SUBSTR,STRING.BIF.PV)
          END
/*
```

Figure 24. Example Job to Specify the Shared Library

Coding your Shared Library Macro Instruction

You must first determine which modules you wish to include in your shared library. The list on the next few pages describes the operands of the PLRSHR macro in terms of the functions of the modules each operand represents. When you have decided which modules will be most frequently used at your site and in which environments, you are ready to code your macro.

Coding Conventions: The coding conventions used in Figure 24 on page 84 and in "Shared Library Macro Operand Descriptions" for coding the PLRSHR macro are as follows:

- Parameters are separated by commas.
- Uppercase letters, numbers, and punctuation marks must be coded exactly as shown.
- Brackets [], and braces {}, must never be coded.
- Lowercase letters and words represent variables for which you must substitute specific information.
- Groups of items within braces {} are related. One of the items must be coded.
- Items or groups of items within brackets [] are optional. They may be omitted at your discretion.

Shared Library Macro Operand Descriptions

The following list describes the operands of the PLRSHR macro instructions. Use it to determine which modules you want to include in your shared library, and then code your macro. See Figure 24 on page 84 for an example of how to code this macro instruction.

Name	Operation	Operand
	PLRSHR	[MODES=(operand[,operand...])] [,ARRAY=(operand[,operand...])] [,CMATH=(operand[,operand...])] [,CONTL=(operand[,operand...])] [,CONV=(operand[,operand...])] [,GRAPHIC=operand] [,IO = (operand[,operand...])] [,JOBSTMT=operand] [,MFUNC=(operand[,operand...])] [,RMATH=(operand[,operand...])] [,STORG=(operand[,operand...])] [,STRGS=(operand[,operand...])]

MODES = ([BASE][,TASK])
specifies whether basic, or tasking, or both basic and tasking shared library modules are required. If both are required, then MODES=(BASE,TASK) must be specified. The macro cannot be used twice to create basic and tasking modules for different function groups, because some routines are

combined into a single module for both tasking and nontasking functions. The IBM-supplied default is MODES=BASE.

ARRAY =
modules associated with array handling and the SUM, PROD, and POLY built-in functions.

BASIC[.LOGIC][.LEAF][.EVENT]

LOGIC
logical operations on arrays ALL and ANY

LEAF
interleaved array module

EVENT
arrays of event variables

FIXED[.PROD][.SUM]

PROD
the PROD built-in function module for fixed binary and fixed decimal elements.

SUM
the SUM built-in function module for fixed decimal elements.

FLOAT[.PROD][.SUM][.POLY]

PROD
the PROD built-in function module for short and long precision floating point element.

SUM
the SUM built-in function module for short and long precision floating point elements.

POLY
the POLY built-in function module for short and long precision floating point elements.

EXTND[.PROD][.SUM][.POLY]

PROD
the PROD built-in function module for extended precision floating point elements.

SUM
the SUM built-in function module for extended precision floating point elements.

POLY
the POLY built-in function module for extended precision floating point elements.

CMATH=

modules associated with COMPLEX variable arithmetic operations and built-in function with COMPLEX variable arguments.

FIXED[.ADD][.MULT][.ABS]

the modules associated with the following built-in functions with complex fixed decimal arguments.

ADD

the module associated with the ADD built-in function

MULT

the module associated with the MULTIPLY built-in function

ABS

the module associated with the ABSOLUTE built-in function

SHORT[.TRIG][.SQRT][.LOG][.ABS][.MULT][.DIV] [.EXPN][.ATRIG]

the modules associated with the following built-in functions with COMPLEX SHORT precision floating point arguments will be made resident.

TRIG

the trigonometric functions SIN, COS, and TAN

SQRT

the square root function

LOG

the LOG and EXP built-in function

ABS

the ABSOLUTE built-in function

MULT

the module associated with the MULTIPLY built-in function

DIV

the module associated with the DIVIDE built-in function

EXPN

the modules associated with exponentiation

ATRIG

the trigonometric function ATAN

LONG[.TRIG][.SQRT][.LOG][.ABS][.MULT][.DIV] [.EXPN][.ATRIG]

the modules associated with the above built-in functions with COMPLEX LONG floating-point arguments will be made resident. (For an explanation of subgroup elements, see CMATH SHORT.)

EXTND[.TRIG][.SQRT][.LOG][.ABS][.MULT][.EXPN][.ATRIG]
the modules associated with the above built-in functions with COMPLEX EXTENDED precision floating-point arguments will be made resident. (For an explanation of subgroup elements, see CMATH SHORT.)

CONTL=
module associated with WAIT, PRIORITY and COMPLETION in a tasking and/or nontasking environment will be made resident.

TASK[.WAIT][.CPLN][.PRTY]
the option is only valid if MODES=(TASK,BASE) has been specified.

WAIT
modules associated with the WAIT statement in a tasking system.

CPLN
modules associated with the COMPLETION pseudovariable in a tasking system.

PRTY
modules associated with the PRIORITY pseudovariable in a tasking system.

NOTASK[.WAIT][.CPLN][.PRTY]
the option is only valid if MODES is not specified and the default MODES=BASE is assumed, or if MODES=BASE or MODES=(BASE,TASK) is specified.

WAIT
modules associated with the WAIT statement in a nontasking system.

CPLN
modules associated with the COMPLETION pseudovariable in a nontasking system.

PRTY
modules associated with the PRIORITY pseudovariable in a nontasking system.

CONV=
specifies that selected parts of the conversion package will be made resident.

ARITH[.CHAR][.DEC][.BIN][.BIT]

CHAR
arithmetic to character conversions.

DEC
arithmetic to decimal conversion.

BIN
arithmetic to binary conversions.

BIT
arithmetic to bit conversions.

EDIT (no subgroup defined)
modules associated with edit stream I/O conversions

BIT[.CHAR][.BIN]

CHAR
bit to character conversions.

BIN
bit to binary number conversions.

CHAR[.ARITH][.BIT][.PIC]

ARITH
character to arithmetic conversions.

BIT
character to bit conversions.

PIC
picture character conversions.

PIC[.CHECK][.DEC][.BIT]

CHECK
checks pictured character and pictured arithmetic input.

DEC
picture decimal conversions

BIT
picture bit conversions

EXTND (no subgroup defined)
converting extended precision floating point variables.

COMPLEX (no subgroup defined)
converting complex variables.

GRAPHIC={YES | NO | (pad,quote,g,ld,rd)}
specifies the graphic padding, left delimiter, right delimiter, graphic quotation mark, and the graphic "G" for graphic string data. An example of specifying the GRAPHIC operand is:

```
GRAPHIC=(4040,427D,42C7,0E,0F)
```

Note: The GRAPHIC options you specify for the shared library *must match* the GRAPHIC options you specified for the compiler.

YES

specifies that the IBM-supplied default options for the graphic padding, graphic quotation mark, graphic "G," left delimiter, and right delimiter will be used.

NO

specifies that no default options are required.

pad

specifies the hexadecimal padding. The IBM-supplied default is X'4040'. The first byte of the padding must be the same as the second byte. For example, X'5050' is valid but X'5060' is not.

quote

is the hexadecimal representation of the graphic quotation mark. The IBM-supplied default is X'427D'.

g

is the hexadecimal representation of the graphic "G." The IBM-supplied default is X'42C7'.

Note: If the preprocessor is used, you may not specify values X'00' through X'06' in either byte of the graphic quotation mark ("quote"option) or graphic "G" ("g" option).

ld

specifies the hexadecimal left delimiter. The IBM-supplied default is X'0E'. You must use X'0E' for the left delimiter, since this is an IBM hardware-generated character.

rd

specifies the hexadecimal right delimiter. The IBM-supplied default is X'0F'. You must use X'0F' for the right delimiter, since this is an IBM hardware-generated character.

Note: If the preprocessor is used, you may not specify values X'00' through X'06' in the left or right delimiter ("ld" or "rd." options). The left or right delimiter also cannot be set to any character in the 60-character set described in *OS and DOS PL/I Language Reference Manual.*

IO =

modules associated with I/O will be made resident.

DATA[.INPUT][.OUTPUT]

INPUT

the modules associated with GET DATA

OUTPUT

the modules associated with PUT DATA

EDIT[.INPUT][.OUTPUT]

INPUT
the modules associated with GET EDIT

OUTPUT
the modules associated with PUT EDIT

LIST[.INPUT][.OUTPUT]

INPUT
the modules associated with GET LIST

OUTPUT
the modules associated with PUT LIST

COPY (no subgroup defined)
the module associated with the COPY option.

RECORD (no subgroup defined)
the module associated with RECORD I/O

EXCL (no subgroup defined)
the module associated with EXCLUSIVE files.

JOBSTMT =
to indicate whether the macro is to generate a JOB statement or whether you will provide one by giving a PUNCH statement for, or placing a REPRO statement immediately preceding, your JOB statement and any continuation lines of the JOB statement.

NOTSUPPLIED (no subgroup defined)
no user-written JOB statement will be in the input stream; a default job statement will be generated.

//PLRSHR JOB (1),'SHARED LIBRARY GEN', MSGLEVEL=1
This option will be the default.

SUPPLIED (no subgroup defined)
you will supply a JOB statement in the input stream.

MFUNC =
specifies that modules associated with various miscellaneous functions will be made resident.

SORT (no subgroup defined)
modules associated with the call of PLISORT

DEBUG[.DUMP][.FLOW]
modules used with debugging aid

DUMP
the modules associated with the call of PLIDUMP.

FLOW
the module associated with the FLOW and COUNT options.

RETC (no subgroup defined)
modules associated with the call of PLIRETC.

CKPT (no subgroup defined)
modules associated with the call of PLICKPT.

DISP (no subgroup defined)
modules associated with the DISPLAY statement.

TIME (no subgroup defined)
modules associated with the TIME built-in function.

DATE (no subgroup defined)
modules associated with the DATE built-in function.

DELAY (no subgroup defined)
modules associated with the DELAY statement.

FETCH (no subgroup defined)
modules associated with dynamic FETCH and RELEASE.

RMATH=
modules used in the REAL arithmetic operations and functions.

FIXED[.ADD][.MULT][.DIV]
the modules associated with the following operations and built-in functions with fixed decimal or binary arguments.

ADD
the module associated with the ADD built-in function

MULT
the module associated with the MULTIPLY built-in function

DIV
the module associated with the DIVIDE built-in function

SHORT[.TRIG][.ATRIG][.HYPER][.AHYPER][.SQRT] [.EXPN][.LOG][.ERF]
the modules associated with the following operations and built-in functions with fixed decimal or fixed binary arguments.

TRIG
the trigonometric functions SIN, COS, TAN, SIND, COSD, TAND.

ATRIG
the trigonometric functions ASIN, ACOS, ATAN, ATAND, ASIND, ACOSD

HYPER
the hyperbolic functions SINH and TANH

ANYPER
the hyperbolic function ATANH

SQRT
the square root function

EXPN
modules associated with exponentiation

LOG
the LOG and EXP built-in functions

ERF
evaluation of the error function

**LONG[.TRIG][.ATRIG][.HYPER][.AHYPER][.SQRT]
[.EXPN][.LOG][.ERF]**
the modules associated with the above built-in functions with LONG floating-point arguments will be made resident. (For an explanation, see RMATH SHORT.)

EXTND[.TRIG][.ATRIG][.HYPER][.AHYPER][.SQRT][.ERF]
the modules associated with the above built-in functions with EXTENDED precision floating-point arguments will be made resident. (For an explanation, see RMATH SHORT.)

STORG=
specifies that storage management and error handling modules will be in the shared library.

ERR[.ONCODE][.ONLOC][.CHECK]

ONCODE
The module associated with the ONCODE built-in function will be made resident.

ONLOC
The module associated with the ONLOC built-in function will be made resident.

CHECK
The module associated with the CHECK condition will be made resident.

PCON[.CTL][.AREA]

CTL
The module associated with allocation and freeing of CONTROLLED variables will be made resident.

AREA
The module associated with allocation and freeing an AREA and assignment between AREAs will be made resident.

STMAP (no subgroup defined)
The modules associated with structure mapping will be made resident.

STRGS=
specifies that string handling modules will be made resident.

BIT[.LOGIC][.COMPARE][.ASSIGN][.INDEX][.REPEAT][.VERIFY][.SUBSTR][.BOOL]
the modules associated with bit built-in functions and logical operations will be made resident.

LOGIC
the logical bit operation modules: AND, OR, and NOT

COMPARE
the modules associated with comparison of bit strings

ASSIGN
the modules associated with assignment of bit strings

INDEX
the bit string built-in function INDEX

REPEAT
the bit string built-in function REPEAT

VERIFY
the bit string built-in function VERIFY

SUBSTR
the bit string built-in function and pseudovariable SUBSTRING.

BOOL
the bit string built-in function BOOL

CHAR[.INDEX][.TRANS][.REPEAT][.VERIFY][.SUBSTR]
the modules associated with the character built-in functions will be made resident.

INDEX
the character string built-in function INDEX

TRANS
the character string built-in function TRANSLATE

REPEAT
the character string built-in function REPEAT

VERIFY
the character string built-in function VERIFY

SUBSTR
the character string built-in function and pseudovariable SUBSTRING

STRING[.BIF][.PV]
the modules associated with the STRING built-in function and pseudovariable will be made resident.

BIF
the STRING built-in function module

PV
the STRING pseudovariable module

Assembling the PLRSHR Macro

When you have selected and specified the groups of modules in the PLRSHR macro instruction, create a job similar to the one in Figure 24 on page 84. The SYSLIB DD statement refers to the shared library system generation macro data set containing the PLRSHR macro added during resident library installation. Run this job to assemble the PLRSHR macro instruction, the output of which is assigned to SYSPUNCH and is the complete job to be run as Job 2. SYSPUNCH may be tape, card, or disk.

Note: When specifying the PLRSHR macro instruction you will select either JOBSTMT=NOTSUPPLIED or SUPPLIED. If your site has specific job statement requirements, you must use the SUPPLIED parameter, and write a PUNCH statement to include the job statement you supply, as shown in Figure 24 on page 84.

If you use the NOTSUPPLIED parameter, the following job statement is automatically generated:

```
//PLRSHR JOB (1),'SHARED LIBRARY GEN',MSGLEVEL=1
```

Running Job 2

Job 2 is the last in the process of creating the shared library. This job will store the shared library in auxiliary storage in the SYS1.PLIBASE or SYS1.PLITASK library or in the equivalent private libraries you have chosen.

Note: If you installed the resident library into private library data sets as we recommended, make sure you change the library names in Job 2 from SYS1.PLIBASE and SYS1.PLITASK to your private library names before you run it.

Job 2 is the SYSPUNCH output from the first job. According to the operand specified for JOBSTMT, the input for Job 2 will either start with the job statement you supplied or a default job statement. Start a reader to the SYSPUNCH data set and run Job 2.

The job will assemble and link-edit the following modules:

- IBMBPSRA onto its basic library, SYS1.PLIBASE, and IBMTPSRA onto the tasking subroutine libraries, SYS1.PLITASK. These will be used subsequently when running PL/I optimizing programs in a shared library environment to resolve subroutine library references to the resident shared library modules.

  Note: During the link-edit of these modules, you will receive the following message from the linkage editor:

  ```
  IEW0461 SYMBOL PRINTED IS AN UNRESOLVED EXTERNAL REFERENCE
  ```

  IEW0461 is issued during the link-edit of the shared library modules. These references will later be resolved when a compiled application program is link-edited.

- IBMBPSMA, IBMBPSLA, and IBMTPSLA, onto SYS1.PLIBASE. These shared library modules will consist of subroutines from the basic or the basic and tasking subroutine libraries, selected according to the options specified in the PLRSHR macro.

The output from the link-edit steps for the above modules gives the storage requirements for the link pack area.

You can test your shared library modules, without making them resident in common storage, by link-editing them into a private library and accessing them with a STEPLIB DD statement.

Loading the Shared Library into Common Storage

Before you can use the shared library, you must load it into MVS common storage, by one of two methods:

- You can include the modules in SYS1.LPALIB, and then perform an initial program load (IPL) specifying the CLPA (create link pack area) option.

  The drawback of this approach is that, if you regenerate your system with the shared library permanently resident in SYS1.LPALIB, you must also rebuild your shared library.

- You can use the MLPA (modified link pack area) facility of MVS. This method causes the shared library modules to be loaded into common storage only until the next initial program load (IPL). Using this method assures that your shared library need not be rebuilt if you replace or upgrade your system.

Using the MLPA Facility

You can place the shared library modules created by job 2 into a private library from which they can be made resident in common storage at IPL time. The procedure is as follows:

1. You must place the modules in a data set contained in the MVS linklist concatenation member in SYS1.PARMLIB (member name LNKLSTxx).

2. You must add the load module names and aliases to an existing MLPA member (IEALPAxx), or create a new MLPA member.

3. When you want to use the shared library, you must perform an initial program load (IPL), specifying the MLPA member that now contains the shared library module names.

The *Initialization and Tuning Guide* for your system contains more information on and examples of creating MLPA members.

Loading the Shared Library for Use under CICS

If you want to use the shared library under CICS, you must specify the PLISHRE=YES option of CICS System Initialization Table (SIT). In addition, you must make the PL/I shared library modules available to CICS. There are two ways to do this:

- Load the shared library modules into MVS common storage using one of the methods described in "Loading the Shared Library into Common Storage" on page 96. This enables batch, CICS, and TSO applications to use the same shared library.

- Place the shared library modules into a private library. If CICS does not find the shared library modules in MVS common storage, it will attempt to load them into its region. In this case, only CICS applications will be able to use the shared library.

For more information on how to use the shared library under CICS, see *OS PL/I Optimizing Compiler: Programmer's Guide.*

Using IBM Cataloged Procedures

You can use IBM-supplied cataloged procedures that use the linkage editor or loader to specify the shared library. This procedure is described in *OS PL/I Optimizing Compiler Programmer's Guide.*

Using the XF Assembler to Create a Shared Library

If you are using the XF Assembler to assemble these modules, it is possible that the total number of external symbol dictionary (ESD) items and/or entry points to be handled in an assembly will exceed the limits of the XF Assembler.

If this occurs, the XF Assembler will produce an error diagnostic during execution of the job stream created by assembling the PLRSHR macro instruction. The PLRSHR macro will also produce a message giving an indication of the number of modules specified and how far the limit has been exceeded. This will cause subsequent assembly and link-edit steps to be flushed, and no shared library modules will be created. In this case, the number of options must be reduced in your PLRSHR macro instruction and the shared library generation process must be repeated.

These problems will not arise when the H Assembler (Program Product Number 5734-AS1) is used.

The following paragraphs describe how to calculate the number of entry points and ESD entries.

The limits of the XF Assembler are:

- The number of external symbol dictionary (ESD) items may not exceed 399.
- The number of ENTRY symbols may not exceed 399.

Figure 25 contains the total number of each of the above items for each PL/I library module subgroup in the shared library. The notes following the figure explain how to calculate the total number of items from Figure 25. Note that the actual numbers for complete functional groups of modules may be less than the totals of figures given for the subgroups contained within them. This applies to the input/output, mathematical, and conversion groups only. Consequently, it might be possible to create a shared library with totals calculated from the table in Figure 25 that slightly exceed the limits of the XF Assembler.

Operand	Subgroups	Column 1	Column 2	Column 3	Column 4
ARRAY	BASIC.LOGIC	3	4	10	4
	BASIC.LEAF	1		1	2
	BASIC.EVENT	2		2	4
	FIXED.PROD	1	1	3	2
	FIXED.SUM	1	1	3	2
	FLOAT.PROD	4	1	6	5
	FLOAT.SUM	4	1	6	5
	FLOAT.POLY	4		4	5
	EXTND.PROD	2	1	4	3
	EXTND.SUM	2	1	4	3
	EXTND.POLY	2		2	3
CMATH	FIXED.ADD	4		4	6
	FIXED.MULT	4		4	6
	FIXED.ABS	2		2	4
	SHORT.SQRT	2		2	4
	SHORT.LOG	14		14	20
	SHORT.TRIG	14		14	20
	SHORT.ATRIG	10		10	14
	SHORT.ABS	2		2	4
	SHORT.MULT	2		2	3
	SHORT.DIV	1		1	2
	SHORT.EXPN	17		17	25
	LONG.SQRT	2		2	4
	LONG.LOG	14		14	20
	LONG.EXPN	17		17	25
	LONG.ABS	2		2	4
	LONG.TRIG	15		15	22
	LONG.ATRIG	10		10	14
	LONG.MULT	2		2	3
	LONG.DIV	1		1	2
	EXTND.SQRT	2		2	4
	EXTND.LOG	14		14	19
	EXTND.TRIG	25		25	31
	EXTND.ATRIG	14		14	18
	EXTND.ABS	3		3	6
	EXTND.MULT	2		2	3
	EXTND.EXPN	8		8	12

Figure 25 (Part 1 of 3). Table of Entry Points and ESD Entries in Shared Library Modules

Operand	Subgroups	Column 1	Column 2	Column 3	Column 4
CONTL	TASK.WAIT	1	2	5	2
	TASK.CPLN	2		2	3
	TASK.PRTY	1		1	2
	NOTASK.WAIT	1	2	5	2
	NOTASK.CPLN	2		2	3
	NOTASK.PRTY	1		1	2
CONV	ARITH.CHAR	1	7	14	2
	ARITH.DEC	15	5	21	17
	ARITH.BIN	9	2	13	11
	ARITH.BIT	3		3	5
	EDIT	4		4	6
	BIT.CHAR	2		2	4
	BIT.BIN	5		5	6
	CHAR.ARITH	13	6	23	17
	CHAR.BIT	1		1	2
	CHAR.PIC	4		4	7
	EXTND	5	15	30	8
	PIC.CHECK	3		3	6
	PIC.DEC	11	8	23	13
	PIC.BIT	1		1	2
	COMPLEX	2	34	43	4
IO	DATA.INPUT	15	11	33	20
	DATA.OUTPUT	19	6	31	24
	LIST.INPUT	13	11	31	17
	LIST.OUTPUT	14	6	26	18
	EDIT.INPUT	24	30	72	35
	EDIT.OUTPUT	26	32	83	39
	RECORD	4		4	4
	EXCL	1		1	1
	COPY	7	4	13	7
MFUNC	SORT	4	2	7	5
	DEBUG.DUMP	1		1	2
	DEBUG.FLOW	3		3	4
	RETC	1		1	2
	CKPT	3		3	4
	DISP	2		2	3
	TIME	1		1	2
	DATE	1		1	2
	DELAY	1		1	2
	FETCH	2		2	3
RMATH	FIXED.ADD	4		4	6
	FIXED.MULT	4		4	7
	FIXED.DIV	4		4	7
	SHORT.TRIG	6		6	8
	SHORT.ATRIG	6		6	9
	SHORT.HYPER	4		4	7
	SHORT.AHYPER	4		4	6
	SHORT.SQRT	1		1	2
	SHORT.EXPN	6		6	10
	SHORT.LOG	4		4	6
	SHORT.ERF	3		3	5
	LONG.TRIG	6		6	8
	LONG.ATRIG	7		7	10
	LONG.HYPER	4		4	7
	LONG.AHYPER	4		4	6
	LONG.SQRT	1		1	2

Figure 25 (Part 2 of 3). Table of Entry Points and ESD Entries in Shared Library Modules

Operand	Subgroups	Column 1	Column 2	Column 3	Column 4
RMATH	LONG.EXPN	6		6	10
	LONG.LOG	4		4	6
	LONG.ERF	3		3	5
	EXTND.TRIG	6		6	8
	EXTND.ATRIG	7		7	10
	EXTND.HYPER	7		7	10
	EXTND.AHYPER	5		5	7
	EXTND.SQRT	1		1	2
	EXTND.ERF	2	1	4	3
STORG	ERR.ONCODE	1		1	2
	ERR.ONLOC	1		1	2
	ERR.CHECK	2	4	9	4
	PCON.CTL	2		2	3
	PCON.AREA	3		3	4
	STMAP	4		4	5
STRGS	BIT.LOGIC	3		3	5
	BIT.COMPARE	1		1	2
	BIT.ASSIGN	6		6	8
	BIT.INDEX	1		1	2
	BIT.REPEAT	6		6	8
	BIT.VERIFY	1		1	2
	BIT.SUBSTR	4		4	5
	BIT.BOOL	4		4	6
	CHAR.INDEX	1		1	2
	CHAR.TRANS	1		1	2
	CHAR.REPEAT	2		2	3
	CHAR.VERIFY	1		1	2
	CHAR.SUBSTR	4		4	5
	STRING.BIF	3	3	8	5
	STRING.PV	2	3	4	4

Figure 25 (Part 3 of 3). Table of Entry Points and ESD Entries in Shared Library Modules

Note that the calculations described below distinguish between the assemblies of modules that are to be assembled for inclusion in the link pack area and those to be included in SYS1.PLIBASE and SYS1.PLITASK. (The former constitute modules IBMBPSL (or IBMTPSL) and IBMBPSM, and will ultimately be loaded into the link pack area; the latter constitute modules IBMBPSR (or IBMTPSR) and will be link-edited into the PL/I load module.

You can make the calculations as follows:

1. To calculate the maximum number of entry points in IBMBPSR (or IBMTPSR), add the figures in column 1 for all subgroups, except the CONTL operand, specified in the PLRSHR macro; to this, add the appropriate tasking or nontasking subgroups of the CONTL operand. Add 4 to this figure to produce the final total, total A.

2. Calculating the number of ESD entries generated in the assemblies (subsequently link-edited together) of module IBMBPSR (or IBMTPSR) requires a calculation for each of the two assemblies.

   To calculate the first total, add the figures in column 2 for the specified subgroups, except CONTL; to this, add the appropriate tasking or nontasking subgroups of the CONTL operand. Add 13 to this to obtain the final total, total B.

To calculate the second total, add the figures in column 4 for the specified subgroups; to this, add the greater of the sums of the tasking or nontasking subgroups of the CONTL operand that are specified. This gives the total C.

3. To calculate the maximum number of entry points generated for IBMBPSM, add the figures in column 2 for the appropriate subgroups (excluding the subgroups of the CONTL operand). This gives the total D.

4. To calculate the maximum number of ESD entries generated for IBMBPSM, add the figures in column 3 for the appropriate subgroups (excluding the subgroups of the CONTL operand). This gives the total E.

The XF Assembler limits will not be exceeded if, for both tasking and nontasking PLRSHR macro instructions, the sum of the maximum of A and D, and the maximum of B, C, and E, is 399 or less. In this case, no further action is required before generating your shared library modules.

If any of the above totals exceed the limits, you should reduce the number of PLRSHR operands specified until the totals are within this amount before proceeding with the shared library generation.

Placing Transient Library Modules in the Link Pack Area

All modules in the transient library, except for CICS-only transient library modules DFHSAP and IBMFnnnn, are eligible to be placed in the link pack area or MLPA. This has the advantage of saving execution time and space when more than one PL/I program is executing concurrently.

You can make the modules resident at any time.

Note: You should, however, be aware of the following facts:

- SMP does not keep track of modules in the link pack area. Therefore, when you apply to the transient library a USERMOD or maintenance that affects a transient library module in the link pack area, you must recopy the updated module to the link pack area.

- If you have programs compiled and link-edited on a previous release of PL/I, and if you had chosen when installing that release to improve initialization time by combining transient initialization/termination modules with resident initialization/termination modules, you must link-edit those programs again with the current resident library to avoid execution-time failures because of a mixture of release levels.

The selection of modules that your site should make resident in the shared area depends upon the type of PL/I programs you normally run.

For example, if your site has a high volume of short compile, link, and go jobs, you could make the transient housekeeping routines resident in the shared area to improve performance.

Figure 26 on page 102 gives a guide to the conditions under which modules could advantageously be made resident. By selecting those conditions that apply to your site, and then considering which of the modules within those sections you would use most frequently, you can build a list of modules to place in the link pack area. Details of how to actually place the modules in the resident area are given under "Using the Shared Library" on page 81. For IBMBPSLA, IBMTPSLA, and IBMBPSMA, substitute the names of the transient library modules to be made resident.

Note: The two housekeeping modules marked with an asterisk (*) in Figure 26 are not needed for programs compiled and link-edited using Release 5.0 and later of OS PL/I. If you are running programs compiled and link-edited using previous releases of OS PL/I and you did *not* combine transient and resident initialization/termination modules, you might want to make these resident at this time.

Condition for Modules to be Resident	**Category**	**Modules**	**Description**	**Approx. Storage Required**
Modules generally needed	House keeping	IBMBPGR	Storage Management (Nonmultitasking)	1690
		IBMBPGD	Storage Management (Nonmultitasking)	2070
		IBMTPGR	Storage Management (Multitasking)	1900
		IBMTPGD	Storage Management (Multitasking)	2590
High volume of short execution	House keeping	IBMBPII*	Program ISA Initialization (Nonmultitasking)	5750
		IBMBPIT*	Program Termination (Nonmultitasking)	650
		IBMTPJI	Program Initialization (Multitasking)	6600
		IBMTPJR	Program Initialization (Multitasking)	4900
		IBMTPIT	Program Termination (Multitasking)	450
		IBMTTEP	ATTACH macro instruction module	16
High volume of OPEN/CLOSE statements or with at least one file being used (especially SYSPRINT)	OPEN/CLOSE	IBMBOCA	Close	3760
		IBMBOPA	Open - Phase 1	1170
		IBMBOPB	Open - Phase 2	3700
		IBMBOPC	Open - Phase 3	3900
		IBMBOPD	Open - Phase 4	1220
		IBMBOPE	Open (VSAM)	3700

Figure 26 (Part 1 of 2). Transient Library Modules for the Link Pack Area

Condition for Modules to be Resident	Category	Modules	Description	Approx. Storage Required
High volume of Stream Input/Output	Stream I/O conversion	IBMBCCL	Conversion Director (Complex strings)	2070
		IBMBCCR	Conversion Director (Noncomplex string)	1150
		IBMBSTA	Tab Table	40
		IBMBSTF	Print File Transmitter	590
		IBMBSTI	Input File Transmitter	380
		IBMBSTV	Print File Transmitter	590
		IBMCSTI	Input File Transmitter (Checkout Compiler Only)	560
		IBMCSTP	Output File Transmitter (Checkout Compiler Only)	1740
Record I/O consecutive (including in-line I/O)	Record I/O consecu-tive	IBMBRQA	Buffered Consecutive Transmitter	1200
		IBMBRQB	Buffered Consecutive Transmitter	1060
		IBMBRQC	Buffered Consecutive Transmitter	1000
Record I/O Indexed	Record I/O Indexed Non-VSAM	IBMBRJA	Indexed Sequential Transmitter	1620
		IBMBRJB	Indexed Sequential Transmitter	1640
		IBMBRKB	Indexed Direct Transmitter	1670
		IBMBRKC	Indexed Direct Transmitter	1810
Record I/O VSAM	Record I/O VSAM	IBMBRVGA	KSDS Sequential Output	1330
		IBMBRVAA	ESDS Transmitter	2160
		IBMBRVHA	KSDS and Path Input/Output	2930
		IBMBRVIA	RRDS Transmitter	2460

Figure 26 (Part 2 of 2). Transient Library Modules for the Link Pack Area

Disabling the PL/I Fast Initialization/Termination Facility

OS PL/I releases prior to release 5.0 provided separate initialization/termination modules that required system LOAD and DELETE functions to perform the initialization tasks. As a post-installation tuning option, you were allowed to combine these modules and thereby decrease the time it took to initialize or terminate the PL/I environment.

Release 5.0 made this feature an integral part of the product. This improves performance but makes program load modules larger. If an application programmer is concerned with the size of these modules and is willing to sacrifice performance, the following example may be used to remove the initialization/termination modules from the application program:

```
//S1         EXEC PLIXCLG,PARM.LKED='LET'
//SYSIN      DD *
  *
  * PL/I application program goes here
  *
//LKED.SYSIN DD *
 REPLACE IBMBPII1,IBMBPIT1,IBMBPGR1
 INCLUDE SYSLIB(IBMBPIRA)
 NAME GO(R)
```

In the above example, the initialization/termination modules will be removed from the load module. This will cause the LINKEDIT step to produce IEW0132 messages. You can ignore these messages.

Index

OS PL/I Optimizing Compiler:
Installation Guide for MVS
SC26-4121-1

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ____________________

Previous TNL ____________________

Previous TNL ____________________

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

SC26-4121-1

Reader's Comment Form

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape

Please do not staple

Fold and tape

IBM®

OS PL/I Optimizing Compiler: Installation Guide for MVS (File No. S370-34) Printed in U.S.A. SC26-4121-1

SC26-4121-1

OS PL/I Optimizing Compiler: Installation Guide for MVS (File No. S370-34) Printed in U.S.A. SC26-4121-1

IBM

SC26-4121-01