SC33-0026-0

Program Product

OS PL/I Optimizing Compiler: System Information

Program Numbers 5734-PL1
5734-LM4
5734-LM5

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

IBM

First Edition (September, 1971)

This edition applies to Release 20.1 of IBM System/360 Operating System and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes will continually be made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 and System/370 Bibliography SRL Newsletter, Order No. GN20-0360, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM United Kingdom Laboratories Ltd., Programming Publications, Hursley Park, Winchester, Hampshire, England. Comments become the property of IBM.

©Copyright International Business Machines Corporation 1971

Order No. SC33-0026-0

Preface

This manual consists of two sections: Storage Estimates and Installation. Each contains system-related information necessary for the installation and use of the OS PL/I Optimizing Compiler, OS PL/I Transient Library, and OS PL/I Resident Library program products that operate under the control of the IBM System/360 Operating System. This manual is designed to be used in conjunction with the following publications:

OS System Generation, Order No. GC28-6554

OS Storage Estimates, Order No. GC28-6551

Each of the two sections of this manual can be inserted in the system manual to which it refers if such an arrangement will simplify your use of documentation.

Prerequisite Publications

Prerequisites for a thorough understanding and for the effective use of this publication are the following publications:

OS Introduction, Order No. GC28-6534

OS System Programmer's Guide, Order No. GC28-6550

In addition to building upon the system publications listed above, this manual refers to the following associated publications:

OS PL/I Optimizing Compiler: Program Logic, Order No. LY33-6007

OS PL/I Transient Library: Program Logic, Order No. LY33-6009

OS PL/I Resident Library: Program Logic, Order No. LY33-6008

OS Job Control Language Reference, Order No. GC28-6704

OS Utilities, Order No. GC28-6586

OS PL/I Optimizing Compiler: Programmer's Guide, Order No. SC33-0006

OS TSO: PL/I Optimizing Compiler, Order No. SC33-0029

AVAILABILITY OF PUBLICATIONS

The availability of a publication is indicated by its use key, i.e., the first letter of its Order Number. The use keys are:

G - General: available to users of IBM systems, products, and services without charge, in quantities to meet their normal requirements; can also be purchased by anyone through IBM branch offices.

S - Sell: can be purchased by anyone through IBM branch offices.

L - Licensed materials, property of IBM: available only to licensees of the related program products under the terms of the license agreement.

Introduction

The optimizing compiler, transient library, and resident library function under the control of the IBM System/360 Operating System, to compile and execute programs written in PL/I.

This publication supplements certain system publications by providing the system-related information which applies particularly to the program products named above. This system-related information includes:

- Compiler, transient library, and resident library storage requirements.
- The procedure needed to add the compiler, transient library, and resident library to the operating system.

The section on "Storage Estimates" defines the amount of core storage and work space for execution required by the compiler, auxiliary storage required by the compiler, transient library, and resident library, and the work space needed for installation.

The "Installation" section should be used by the system programmer or planner responsible for system generation and maintenance. It discusses the system requirements for using the compiler, transient library, and resident library and describes how to add them to an operating system. This section also includes information on how to tailor the compiler and resident library to installation requirements.

Note: The transient library used with the optimizing compiler is identical to that used with the checkout compiler. The installation procedure for the transient library is therefore the same for use with the optimizing and checkout compilers.

OS
PL/I Optimizing Compiler:
Storage Estimates

© Copyright International Business Machines Corporation 1971

Contents

PL/I Optimizing Compiler Storage Requirements 1
  Core Storage Requirements 1
  Auxiliary Storage Requirements 1
  Compiler Work Space Requirements . . . 2
  Work Space Required During Installation of Compiler 3
    Work File Space Requirements 3
    SYS1.PPOPTION Space Requirements (user system generation macro options) 4
PL/I Transient Library Storage Requirements 4
  Auxiliary Storage Requirements 4
  Work Space Required During Installation of Transient Library . . 5
    Work File Space Requirements 5
PL/I Resident Library Storage Requirements 5
  Auxiliary Storage Requirements 5
  Work Space Required During Installation of Resident Library . . . 7
    Work File Space Requirements 7
    SYS1.PPOPTION Space Requirements (User System Generation macro options) 7

Figures

Figure 1. Minimum core storage requirements 1
Figure 2. Link library storage requirements (compiler) 2
Figure 3. Subroutine library storage requirements (compiler) 2
Figure 4. Procedure library storage requirements (compiler) 3
Figure 5. Work space requirements for the compiler 3
Figure 6. Installation work file space requirements (compiler) 4
Figure 7. SYS1.PPOPTION space requirements 4
Figure 8. Link library storage requirements (transient library) 5
Figure 9. Installation work file space requirements (transient library) . 5
Figure 10. Subroutine library storage requirements (resident library) 6
Figure 11. Procedure library storage requirements (resident library) 6
Figure 12. Installation work file space requirements (resident library) . 7
Figure 13. SYS1.PPOPTION space requirements 7
Figure 14. (Part 1 of 3). Link library storage requirements (shared library feature) 8
Figure 14. (Part 2 of 3). Link library storage requirements (shared library feature) 9
Figure 14. (Part 3 of 3). Link library storage requirements (shared library feature) 10

Storage Estimates

The storage estimates given in this section are intended to supplement the system figures given in the publication: OS Storage Estimates. When you have determined the amount of storage needed for the operating system as described in that manual, use the figures given here to determine the additional amounts needed for the PL/I Optimizing Compiler, PL/I Transient Library, and PL/I Resident Library program products.

Estimates are given for the amount of core storage required for compiler execution, the amount of auxiliary storage required on a system library for the compiler, transient library, and resident library, the work space required during installation, and the amount of additional work space needed for compilation.

If you require additional information about the modules that make up the compiler, transient library, and resident library program products, please refer to the publications: OS PL/I Optimizing Compiler: Program Logic, OS PL/I Transient Library: Program Logic, and OS PL/I Resident Library: Program Logic.

PL/I Optimizing Compiler Storage Requirements

Figures 1, 2, 3, and 4 give the storage requirements and figure 5 the work space requirements for the PL/I Optimizing Compiler.

CORE STORAGE REQUIREMENTS

Figure 1 defines the minimum core storage requirements for the compiler. The estimates include the requirements for access methods used by the program. If the access methods are resident, the requirement can be reduced by the sum of the resident modules.

Information on execution storage requirements for a particular PL/I program can be found by using the REPORT option. This is described in the publication: OS PL/I Optimizing Compiler: Programmer's Guide.

AUXILIARY STORAGE REQUIREMENTS

The auxiliary storage required in system libraries by the compiler is given in figures 2, 3, and 4.

Note: Link library storage estimates refer to space requirements in either an existing system link library data set, or a new private link library data set. The choice of which type of data set to use is discussed in the "Installation" section under the heading "Preparing for Installation (General)."

Processing program	Access method used	Storage requirement (in bytes)
PL/I Optimizing Compiler (MFT)	BSAM, QSAM	51,200
PL/I Optimizing Compiler (MVT)	BSAM, QSAM	53,248

Figure 1. Minimum core storage requirements

Description	No. of directory blocks(1)	Number of tracks required							
		IBM 2301	IBM 2302	IBM 2303	IBM 2311	IBM 2314	IBM 3330	IBM 2305	IBM 2321
PL/I Optimizing Compiler	11	85	309	353	425	236	128	153	775

(1) The number of 256-byte directory blocks to be allocated for a directory when a partitioned data set is being defined. The number of directory blocks that can be contained on a track is as follows:

IBM 2301 - 45 directory blocks/track
IBM 2302 - 14 directory blocks/track
IBM 2303 - 12 directory blocks/track
IBM 2311 - 10 directory blocks/track
IBM 2314 - 17 directory blocks/track
IBM 3330 - 75 directory blocks/track
IBM 2305 - 90 directory blocks/track
IBM 2321 - 5 directory blocks/track

Example

Total space for the PL/I Optimizing Compiler on 2311 disk storage is:

$T = T_D + T_C$ where T_D = space for directory blocks (tracks) and is calculated as follows:

$T = \frac{11}{10} + 425$

$T_D = \frac{D}{N}$

where D = number of directory blocks.
N = number of directory blocks/track.

Total space = 427 tracks
Note: All figures should be rounded up to the next integral number of tracks.

where T_C = space for compiler modules (tracks)

Figure 2. Link library storage requirements (compiler)

Description	No of directory blocks	Number of tracks required							
		IBM 2301	IBM 2302	IBM 2303	IBM 2311	IBM 2314	IBM 3330	IBM 2305	IBM 2321
TSO command processor (optional)	1	2	9	10	11	6	4	4	19
TSO HELP module (optional)	1	2	13	15	17	9	5	6	32

Figure 3. Subroutine library storage requirements (compiler)

COMPILER WORK SPACE REQUIREMENTS

The compiler requires additional work space beyond the core storage needed for execution. This work space requirement varies according to the number of source cards or the amount of available main storage, or both. Figure 5 estimates the work space the compiler might require to process typical source programs. According to the type and combination of statements involved, however, the storage requirements may vary widely.

Description	Number of tracks required							
	IBM 2301	IBM 2302	IBM 2303	IBM 2311	IBM 2314	IBM 3330	IBM 2305	IBM 2321
IBM cataloged procedures and LOGON procedure distributed with PL/I Optimizing Compiler	1	3	3	3	2	1	2	5

Note: 1. The number of directory blocks is 1. If the compiler and resident library cataloged procedures are on the same data set the number of directory blocks is 1 for the combined set. (When allocating space for SYS1.PROCLIB, the number of 256-byte directory blocks to be allocated for a directory must be indicated in the SPACE parameter. See the description of the SPACE parameter of the DD statement in the publication: OS Job Control Language Reference.)
2. The space requirements assume maximum values and an unblocked data set (i.e., BLKSIZE=80).

Figure 4. Procedure library storage requirements (compiler)

Data set	No. of source state-ments	PL/I Optimizing Compiler operating in	Number of tracks required							
			IBM 2301	IBM 2302	IBM 2303	IBM 2311	IBM 2314	IBM 3330	IBM 2305	IBM 2321
SYSUT1	150	50K	6	23	26	30	16	9	11	60
		100K	6	23	26	30	16	9	11	60
		200K	2	8	9	10	6	4	4	20
	500	50K	19	82	95	110	57	32	38	220
		100K	19	82	95	110	57	32	38	220
		200K	9	38	44	50	26	15	18	100
	1000	50K	42	179	206	240	124	69	83	480
		100K	38	164	189	220	114	64	76	440
		200K	35	149	172	200	103	58	69	400

Note:
- These estimates are based on the assumption that the input is 80-character records.
- These estimates are derived from a set of sample PL/I programs, however the actual values may vary very widely, infact by as much as +100% to -50%, depending on the complexity of the statements in the program.
- Certain options will cause increases in the size of the spill file. For example OPT(TIME) will possibly cause a 50% increase, and the 48 character set will also increase the spill file. The options ATR, XREF, and LIST will not increase the spill file size significantly.

Figure 5. Work space requirements for the compiler

WORK SPACE REQUIRED DURING INSTALLATION OF COMPILER

Work File Space Requirements

Work files used during installation require space on public volumes as follows:

	Device type	Unit name used	Space (tracks) 2311	Space (tracks) 2314	Comments
OS PL/I Optimzing Compiler	Two 9 track tapes or DASD	SYSSQ	80	40	Space for two assembler work files
	DASD	SYSDA	60	30	One assembler work file per linkage editor work file
	DASD	SYSDA	5	2	Intermediate assembler output
	DASD	SYSDA	40	20	Must be on the same device as SYS1.PPOPTION
MFT/MVT DD * and SYSOUT Space Direct access space equivalent to 900 2311 tracks is required for DD * data sets, and 250 2311 tracks for SYSOUT data sets.					

Figure 6. Installation work file space requirements (compiler)

SYS1.PPOPTION Space Requirements (user system generation macro options)

	Space (tracks) 2311	Space (tracks) 2314	Directory blocks *
OS PL/I Optimizing Compiler	5	3	1
* Only one directory block required for both compiler and resident library together.			

Figure 7. SYS1.PPOPTION space requirements

PL/I Transient Library Storage Requirements

AUXILIARY STORAGE REQUIREMENTS

The auxiliary storage required in system libraries by the transient library is given in figure 8.

Description	No. of directory records (1)	Number of tracks required							
		IBM 2301	IBM 2302	IBM 2303	IBM 2311	IBM 2314	IBM 3330	IBM 2305	IBM 2321
PL/I Transient Library	20	11	45	52	60	31	18	21	124

(1) The number of 256-byte directory records to be allocated for a directory when a partitioned data set is being defined.

Note: For the number of directory records per track, and an example of calculating the space required, refer to figure 2.

Figure 8. Link library storage requirements (transient library)

WORK SPACE REQUIRED DURING INSTALLATION OF TRANSIENT LIBRARY

Work File Space Requirements

The work file used during installation requires space on a public volume as shown in figure 9.

	Device type	Unit name used	Space (tracks)		Comments
			2311	2314	
OS PL/I Transient Library	DASD	SYSDA	60	30	One linkage editor work file

MFT/MVT DD * and SYSOUT Space

Direct access space equivalent to 180 2311 tracks is required for DD * data sets, and 40 2311 tracks for SYSOUT data sets.

Figure 9. Installation work file space requirements (transient library)

PL/I Resident Library Storage Requirements

Storage requirements for the resident library and the shared library (if installed) are shown below.

AUXILIARY STORAGE REQUIREMENTS

The auxiliary storage required in system libraries by the resident library is shown in figures 10 and 11, and for the shared library feature in figure 14.

Description	No. of directory blocks	Number of tracks required							
		IBM 2301	IBM 2302	IBM 2303	IBM 2311	IBM 2314	IBM 3330	IBM 2305	IBM 2321
SYS1.PLIBASE (non-tasking) subroutine library:									
Basic modules	24	6	26	30	35	18	10	12	62
Complex function modules (optional)	7	2	8	9	10	6	3	4	21
Extended precision floating point modules (optional)	3	2	6	7	8	5	3	3	16
Interlanguage communication modules (optional)	1	1	1	1	1	1	1	1	2
SYS1.PLITASK subroutine library:									
Tasking modules (optional)	2	1	3	4	4	3	2	2	8

Note: The total space needed for SYS1.PLIBASE (main non-tasking subroutine library) is that for the basic modules, plus that for any optional module groups you choose. Additional space in SYS1.PLIBASE and/or SYS1.PLITASK is needed if you install the Shared Library feature. This is detailed in figure 14.

Figure 10. Subroutine library storage requirements (resident library)

Description	Number of tracks required							
	IBM 2301	IBM 2302	IBM 2303	IBM 2311	IBM 2314	IBM 3330	IBM 2305	IBM 2321
IBM cataloged procedures distributed with PL/I Resident Library.	1	1	1	1	1	1	1	2
Private macro library data set for shared library PLRSHR system generation macro	21	90	103	120	62	35	42	209

Note:
- The number of directory blocks is 1.
- See the note on the SPACE parameter in figure 4. (IBM supplied cataloged procedure only.)
- Space requirements assume a data set blocksize of 3440 bytes. (Macro library data set only.)

Figure 11. Procedure library storage requirements (resident library)

WORK SPACE REQUIRED DURING INSTALLATION OF
RESIDENT LIBRARY

Work File Space Requirements

Work files used during installation require
space on public volumes as follows:

	Device type	Unit name used	Space (tracks) 2311	Space (tracks) 2314	Comments
OS PL/I Resident Library	Two 9 track tapes or DASD	SYSSQ	80	40	Space for two assembler work files
	DASD	SYSDA	60	30	One assembler work file per linkage editor work file
	DASD	SYSDA	40	20	Must be on the same device as SYS1.PPOPTION
MFT/MVT DD * and SYSOUT Space Direct access space equivalent to 400 2311 tracks is required for DD * data sets, and 75 2311 tracks for SYSOUT data sets.					

Figure 12. Installation work file space requirements (resident library)

SYS1.PPOPTION Space Requirements (User
System Generation macro options)

	Space (tracks) 2311	Space (tracks) 2314	Directory blocks *
OS PL/I Resident Library	1	1	1
*Only one directory block required for both compiler and resident library together.			

Figure 13. SYS1.PPOPTION space requirements

Module	Parameter	Sub-parameter	Storage Requirement (bytes)
IBMBPSLA & IBMTPSLA	MODES=	TASK	5860
		BASE	3010
	CONTL=	TASK.WAIT	1024
		TASK.CPLN	140
		TASK.PRTY	190
		NOTASK.WAIT	890
		NOTASK.CPLN	160
		NOTASK.PRTY	120
IBMBPSMA	ARRAY=	BASIC.LOGIC	510
		BASIC.LEAF	110
		BASIC.EVENT	590
		FIXED.PROD	640
		FIXED.SUM	490
		FLOAT.PROD	440
		FLOAT.SUM	400
		FLOAT.POLY	400
		EXTND.PROD	440
		EXTND.SUM	360
		EXTND.POLY	450
	CMATH=	FIXED.ADD	670
		FIXED.MULT	1010
		FIXED ABS	770
		SHORT.SQRT	480
		SHORT.LOG	1630
		SHORT.TRIG	1610
		SHORT.ATRIG	1150
		SHORT.ABS	310
		SHORT.MULT	130
		SHORT.DIV	110
		SHORT.EXPN	2350
		LONG.SQRT	480
		LONG.LOG	2100
		LONG.EXPN	2830
		LONG.ABS	310
		LONG.TRIG	2540
		LONG.ATRIG	1430
		LONG.MULT	130
		LONG.DIV	110
		EXTND.SQRT	690
		EXTND.LOG	4320
		EXTND.TRIG	6260
		EXTND.ATRIG	3960
		EXTND.ABS	870
		EXTND.MULT	550
		EXTND.EXPN	3000
	CONV=	ARITH.CHAR	840
		ARITH.DEC	980
		ARITH.BIN	1040
		ARITH.BIT	820
		EDIT	1160
		BIT.CHAR	620
		BIT.BIN	500
		CHAR.ARITH	2690
		CHAR.BIT	430
		CHAR.PIC	1230

Figure 14. (Part 1 of 3). Link library storage requirements (shared library feature)

Module	Parameter	Sub-parameter	Storage Requirement (bytes)
IBMBPSMA (continued)		ETND	5130
		PIC.CHECK	1620
		PIC.DEC	2060
		PIC.BIT	260
		COMPLEX	1240
	IO=	DATA.INPUT	6280
		DATA.OUTPUT	4860
		LIST.INPUT	4040
		LIST.OUTPUT	3600
		EDIT.INPUT	6240
		EDIT.OUTPUT	6320
		RECORD	170
		EXCL	60
		COPY	980
	JOBSTMT=	NOTSUPPLIED	-
		SUPPLIED	-
	MFUNC=	SORT	1650
		DEBUG.DUMP	240
		DEBUG.FLOW	1420
		RETC	50
		CHKPT	580
		DISP	760
		TIME	110
		DATE	150
		DELAY	110
		FETCH	200
	RMATH=	FIXED.ADD	670
		FIXED.MULT	1000
		FIXED.DIV	1460
		SHORT.TRIG	600
		SHORT.ATRIG	810
		SHORT.HYPER	640
		SHORT.AHYPER	460
		SHORT.SQRT	180
		SHORT.EXPN	840
		SHORT.LOG	540
		SHORT.ERF	680
		LONG.TRIG	740
		LONG.ATRIG	1020
		LONG.HYPER	970
		LONG.AHYPER	630
		LONG.SQRT	170
		LONG.EXPN	1130
		LONG.LOG	820
		LONG.ERF	1160
		EXTND.TRIG	1710
		EXTND.ATRIG	2260
		EXTND.HYPER	2980
		EXTND.AHYPER	2400
		EXTND.SQRT	330
		EXTND.ERF	1450
	STORG=	ERR.ONCODE	230
		ERR.ONLOC	160
		ERR.CHECK	640

Figure 14. (Part 2 of 3). Link library storage requirements (shared library feature)

Module	Parameter	Sub-parameter	Storage Requirement (bytes)
IBMBPSMA (continued)		PCON.CTL	160
		PCON.AREA	560
		STMAP	1790
	STRGS=	BIT.LOGIC	980
		BIT.COMPARE	250
		BIT.ASSIGN	1350
		BIT.INDEX	360
		BIT.REPEAT	1350
		BIT.VERIFY	220
		BIT.SUBSTR	360
		BIT.BOOL	1080
		CHAR.INDEX	220
		CHAR.TRANS	820
		CHAR.REPEAT	660
		CHAR.VERIFY	220
		CHAR.SUBSTR	360
		STRING.BIF	1830
		STRING.PV	1400

- To use this table; locate your subparameters, and add the storage requirements.
- If you specify a combination of subparameters, add the storage requirements individually.
- Storage is required, on SYS1.LINKLIB, for the modules selected with this option. Convert the storage required into tracks by using the following conversion factors:

IBM 2301 Drum storage device	6×10^{-5} tracks/byte
IBM 2302 Disk storage device	26×10^{-5} tracks/byte
IBM 2303 Drum storage device	30×10^{-5} tracks/byte
IBM 2311 Disk storage device	35×10^{-5} tracks/byte
IBM 2314 Disk storage device	18×10^{-5} tracks/byte
IBM 3330 Disk storage device	10×10^{-5} tracks/byte
IBM 2305 Disk storage device	12×10^{-5} tracks/byte

EXAMPLE: If the PLRSHR macro shared library feature is specified as
MODES=TASK,CONV=(EDIT,BIT,CHAR),STRGS=BIT
the storage requirement is:

```
 5860      (basic requirement)
+1160
+1120
+4350
+5950
-----
18,440     bytes
```

18,440 bytes (35×10^{-5} tracks/byte) = 7 tracks on a 2311

Figure 14. (Part 3 of 3). Link library storage requirements (shared library feature)

OS
PL/I Optimizing Compiler:
Installation

© Copyright International Business Machines Corporation 1971

Contents

System Requirements 1
Machine Requirements 1
Operating System Requirements 1
Program Product Distribution 2
Installation Process 2
Description of Job Stream Content . . 2
Using Installation Tapes 3
Modifying the START RDR Command . . 3
Unblocking the Installation Tapes . 4
Multiple Program Product Installation 4
Preparing for Installation (General) . . 4
Preparing for Compiler Installation . . 5
Allocating Space for the Compiler . . 5
Creating the Option Data Set 8
Installing Compiler 9
Starting Reader to Tape 9
Example 1 - User Specified JCL for the Compiler 9
Sample Program 12
Sample Program Output 12
Default Compile-time Options 12
Preparing for Transient Library Installation 22
Allocating Space for the Transient Library 22
Installing Transient Library 22
Starting Reader to Tape 22
Example 2 - User Specified Jcl For The Transient Library 22
Preparing for Resident Library Installation 24
Allocating Space for the Resident Library 24
Creating the Option Data Set 26
Resident Library Options 26
Installing Resident Library 26
Starting Reader to Tape 26
Example 3 - User Specified Jcl For The Resident Library 27
Shared Library 29
Creating The Shared Library 29
Shared Library in Main Storage . . . 30
Initial Program Load (IPL) 30
Shared Library System Generation Macro 31

Figures

Figure 1. Preparation and installation (compiler) 6
Figure 2. (Part 1 of 2). Example of user specified JCL - compiler 10
Figure 2. (Part 2 of 2). Example of user specified JCL - compiler 11
Figure 3. (Part 1 of 3). Default compile-time options 13
Figure 3. (Part 2 of 3). Default compile-time options 14
Figure 3. (Part 3 of 3). Default compile-time options 15
Figure 4. Preparation and installation (transient library) 23
Figure 5. Example of user specified JCL - transient library 24
Figure 6. Preparation and installation (resident library) 25
Figure 7. Resident library options . . 26
Figure 8. (Part 1 of 2). Example of user specified JCL - resident library . 28
Figure 8. (Part 2 of 2). Example of user specified JCL - resident library . 29
Figure 9. Example job to specify the shared library. 30
Figure 10. (Part 1 of 2). Compiler job stream 37
Figure 10. (Part 2 of 2). Compiler job stream 39
Figure 11. Transient library job stream 41
Figure 12. (Part 1 of 3). Resident library job stream 43
Figure 12. (Part 2 of 3). Resident library job stream 45
Figure 12. (Part 3 of 3). Resident library job stream 47

Installation

This section begins by defining the minimum system requirements for installing and using the optimizing compiler under the control of the IBM System/360 Operating System. It then describes the installation procedure for adding the compiler, transient library, and resident library to an existing MFT or MVT operating system.

Before the program products can be added to any operating system, the operating system must be generated to meet the minimum system requirements. For information on operating system generation, see the publication: OS System Generation.

System Requirements

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

MACHINE REQUIREMENTS

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

- An IBM System/360 with a minimum of 128K bytes of main storage of which at least a 50K byte partition for MFT and a 52K byte partition for MVT must be available to the compiler, or a 72K byte partition when using TSO.
- Floating point and decimal instruction sets.
- Interval timer, if PL/I TIME built-in function or the DELAY statement are to be used or compilation times are to be printed out by the compiler.
- A device, such as a card reader, direct access storage device, or magnetic tape unit, for the job input stream.
- The IBM 1050 printer-keyboard, or equivalent device.
- A printer, or magnetic tape unit, for the system output file.
- A direct access device for work files and program product residence.

In addition an IBM 2400 magnetic tape drive is needed during installation to process the distribution tape(s).

OPERATING SYSTEM REQUIREMENTS

An MFT or MVT Operating System, including the following:

- MVT TSO option for conversational invocation of the compiler.
- Time of day support if the PL/I TIME built-in function is to be used.
- Interval/Job step timing support if the PL/I DELAY statement is to be used.
- Operator communication facility at IPL time if the resident library shared library feature is to be generated.
- An eligible automatic restart user abend code of 4092 if PL/I Checkpoint/Restart interface is to be used, and automatic restarts to be forced by PL/I programs during execution using the PLIREST statement.
- BDAM, ISAM, and TCAM if PL/I direct, indexed, and teleprocessing support is to be used at execution time.
- An F-level linkage editor.
- OS SORT if sort/merge interfaces are to be used at execution time ; OS FORTRAN (E), (G), or (H) if communicating with FORTRAN programs; and OS COBOL (E), (F), or American National Standard COBOL if communicating with COBOL programs.

In addition, the following system features are required to run the installation job streams for all three program products:

- An F-level linkage editor, with the alias name IEWL.
- Generic unit names of SYSSQ and SYSDA meaning any sequential input/output device (tape or direct access), and any direct access device, respectively.

  Note: Only SYSDA is required for installing the transient library.

- Use of SYSOUT=A and SYSOUT=B for printed and punched output respectively.

  Note: Only SYSOUT=A is used by the transient library job stream.

For the compiler and resident library, an F-level (or larger) assembler, with the alias name ASMBLR, is needed to process the system generation macros used to tailor these products.

Program Product Distribution

The PL/I Optimizing Compiler, PL/I Transient Library, and PL/I Resident Library are distributed in the form of OS job streams on magnetic tape(s), one job stream per program product.

The installation tape will be one of the following forms:

- 9-track, unlabeled, 800 bpi.
- 9-track, unlabeled, 1600 bpi.
- 7-track, unlabeled, 800 bpi.

The job streams on tape consist of blocked 80-byte card images with a block size of 3440 bytes.

Distribution of the PL/I Optimizing Compiler and its associated libraries is done in two ways, either of which you may choose. These are:

- The compiler, transient library, and resident library program products on separate tapes.
- The compiler, transient library, and resident library program products (in that order) on the same tape.

The second system of distribution consists of three job streams formed as three files on tape, one file per program product.

Installation Process

Installation of the compiler, transient library, and resident library can be done at any time after your operating system is initially generated.

You do this by running the job streams for each program product under the control of your operating system, having first defined and pre-allocated the data sets, or target libraries, into which they will be added.

Provision is made in the compiler and resident library job streams for you to tailor the program products to suit your installation's needs, by specifying options and/or variations in OS Assembler macros of the type used to generate your operating system.

The compiler, transient library, and resident library job streams are not inter-dependent, and the program products may be installed separately, in any order. Considerations for parallel installation are discussed later in "Multiple Program Product Installation."

Note: The distribution package prepared by IBM should be retained as backup in case the operating system has to be re-created or you wish to change the defaults for the compile-time options.

DESCRIPTION OF JOB STREAM CONTENT

Figures 10, 11, and 12, at the end of this section, give a brief description of the jobs accomplished by the compiler, transient library, and resident library job streams when run, the JCL statements contained in these job streams, and a graphic summary of the installation process.

The following actions are accomplished when the job streams are run:

Compiler (also see figure 10)

1. The modules comprising the compiler are linkage edited into a link library data set named by you.
2. A default compile-time options module is generated, with values corresponding to those specified by you in a system generation macro.
3. A TSO command processor and HELP modules (optional) are linkage edited to subroutine library data sets named by you.
4. IBM supplied cataloged procedures and a LOGON procedure (optional), are added to a procedure library data set specified by you.
5. A sample program for batch mode is punched down on cards, which you can

run both to check satisfactory installation, and also to illustrate use of the compiler. The output from this program may subsequently be used with the resident and transient libraries.

Transient Library (also see figure 11)

The modules comprising the transient library are linkage edited to a link library data set named by you.

Resident Library (also see figure 12)

1. The basic modules comprising the resident library are linkage edited to a subroutine data set named by you (normally SYS1.PLIBASE).

2. The following optional groups of modules are linkage edited to SYS1.PLIBASE if you request their inclusion in a system generation macro:

   - Complex function subroutines
   - Interlanguage communication subroutines
   - Extended precision floating point subroutines
   - Complex extended float arithmetic modules
   - Tasking library

3. Tasking modules are optionally linkage edited to a different subroutine data set specified by you (normally SYS1.PLITASK). You request their inclusion in the same system generation macro specification as that to request the other optional module groups.

4. IBM supplied cataloged procedures, for use in linkage editing PL/I programs, are added to a procedure library data set specified by you.

5. A special system generation macro is added to a private macro library data set. This macro may be used, after the resident library has been installed, to generate the resident library shared library feature. Its use is discussed later under the heading "Shared Library."

USING INSTALLATION TAPES

The tapes containing the installation job streams distributed for the compiler, transient library, and resident library are blocked, with a block size of 3440 bytes and logical record length of 80 bytes. You must allow for this when you run the job streams on your operating system. You can do this by either modifying the normal START RDR command used to read the tapes, or by unblocking the tapes by copying them to larger tapes.

Modifying the START RDR Command

Using the standard IBM supplied reader procedure, enter the following command from the system console device:

```
START RDR,cuu,,,DCB=(RECFM=FB,
    BLKSIZE=3440,BUFL=3440,DEN=d)
```

where: cuu = address of the magnetic tape unit used (c=channel, uu=unit address, for example 182)

d = density of installation tape (d=2 for 800 b.p.i. tapes, d=3 for 9 track 1600 b.p.i. tapes).

If you have ordered the three products on a single tape, use the above command for the first file (compiler job stream), but specify the file number for the second and third files (transient and resident library job streams, respectively), as follows:

```
START RDR,cuu,,,DCB=(RECFM=FB,
    BLKSIZE=3440,BUFL=3440,
    DEN=d),LABEL=(n,NL)
```

where: n = 2 for the transient library job stream

n = 3 for the resident library job stream

Note:

On MFT you will need a reader partition of at least 50K bytes. The size needed may be larger if your installation has a blocked cataloged procedure library, or your reader procedure has been modified to use multiple buffers or produce blocked output. The actual size you need can be obtained from the publication: OS Storage Estimates.

On MVT your reader procedure must specify a region size of at least 52K

bytes, unless reader modules are permanently resident. As for MFT, you may need a larger region size if your installation has a blocked procedure library, or your reader procedure has been modified as described for MFT above.

If your reader procedure does not specify a large enough region size, and you do not want to unblock the tapes, you will have to add a modified reader procedure to your procedure library, specifying a suitable region size (and modified input DCB parameters if you wish). The publications: OS Storage Estimates and OS System Programmer's Guide tell you how to calculate the region size needed, and how to modify reader procedures.

Unblocking the Installation Tapes

If you want to use your normal reader procedure to run the job streams, and you cannot use the method suggested above, you must first unblock the installation tapes by copying them to larger volumes. You can do this using the IEBGENER utility as shown in the JCL below.

where: mmmm (for the input tape) and nnnn (for the output tape) = address or generic name of magnetic tape device (for example, 2400-2 for 7 track; 2400 for 9 track, etc.)

n = number of file being copied. (n=1 can be omitted for the first or only file on tape; n=2 for transient library on combined installation tape; n=3 for resident library on combined installation tape.)

d = density of tape (d=2 for 800 b.p.i. tapes, d=3 for 1600 b.p.i. tapes).

The volume serial parameters serve only as identification values in operator tape mounting messages, since the tapes are unlabeled. When unblocking, the size of the output tapes you require are as follows:

Compiler job stream,
or all three
jobs on one tape ------- 1 x 2400 foot tape

Transient library
job stream ------------- 1 x 1200 foot tape

Resident library
job stream ------------- 1 x 1200 foot tape

```
//UNBLOCK   JOB   ---
//COPY      EXEC  PGM=IEBGENER
//SYSUT1    DD    UNIT=mmmm,VOL=SER=BLKD,DISP=OLD,LABEL=(n,NL),
//            DCB=(RECFM=FB,BLKSIZE=3440,LRECL=80,DEN=d)
//SYSUT2    DD    UNIT=nnnn,VOL=SER=UNBLKD,LABEL=(n,NL),
//            DCB=(RECFM=F,BLKSIZE=80,LRECL=80,DEN=d)
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    DUMMY
```

MULTIPLE PROGRAM PRODUCT INSTALLATION

Attempting to install the compiler, transient library, and resident library at the same time in a multiprogramming environment may, during installation, cause problems due to data set contention. You can either:

1. Choose unique libraries for each target data set in all program product installation procedures, which will eliminate data set contention, and multiprogramming will be achieved.

2. Prevent unnecessary enqueuing of jobs on a single data set by starting just one initiator to Job CLASS A (default Job CLASS used in distribution job-stream). This would force sequential processing of program product applications, but allow maximum multiprogramming with your normal jobs.

Preparing for Installation (General)

The tape(s) contains the object modules that make up the compiler, transient library, and resident library as well as the job control language statements needed to add them to your system. Before the tape(s) can be used, you must define and allocate space for the partitioned data sets in which the compiler, transient

library, and resident library are to reside. These data sets are target libraries. The target libraries can be the link library (SYS1.LINKLIB), procedure library (SYS1.PROCLIB), subroutine libraries (normally SYS1.PLIBASE and SYS1.PLITASK), or any other library names you select. For the link and procedure libraries, it is highly desirable to choose separate data sets from those used to contain your operating system modules, from now on referred to as private libraries. This means, that if some time in the future you decide to generate a new version of your operating system, it will be a relatively easy procedure to move your program products to the new system.

If you choose private libraries, it will be necessary for you to take some additional action upon completion of the installation procedure, before you use the program product. If you choose a private link library data set, then it must be concatenated to SYS1.LINKLIB using the OS Link Library List option. If you choose a private procedure library, the modules can either be moved to SYS1.PROCLIB, or a special reader procedure used to point to the private library. (In both cases see the publication: OS System Programmer's Guide, for further details.) The subroutine libraries require no further user action, although the IBM supplied cataloged procedures will need changing if a name other than SYS1.PLIBASE is used.

For the compiler only, you can optionally generate a TSO command processor (prompter), and a TSO HELP module, in a TSO command processor and HELP library respectively. They may be the system libraries, SYS1.CMDLIB and SYS1.HELP, or private libraries. Whichever you choose, the factors to be considered are the same as those discussed earlier for link and procedure libraries, except that use of private TSO libraries may adversely affect TSO system response time.

If you use a private command processor library, after compiler installation, this must be concatenated to SYS1.CMDLIB in STEPLIB DD statements within your LOGON procedure for the optimizing compiler. If you use a private HELP library, then it must be concatenated with SYS1.HELP before the compiler is used after installation.

For the compiler and resident library only, you must also catalog and allocate a partitioned data set named SYS1.PPOPTION. The job streams read macro definitions, specified by you, from this data set, and use them to tailor the products to suit your installation's requirements.

The job streams are not interdependent, and you can do the preparation steps for all three products before installing them, or do the preparation and installation for each program product in turn as shown in Examples 1, 2 and 3 later in this section.

Details of preparation and installation are described separately for each of the compiler, transient library, and resident library program products. Wherever the procedures are the same, the main description is contained in the section covering the compiler, and the corresponding sections for the other two products make reference to it.

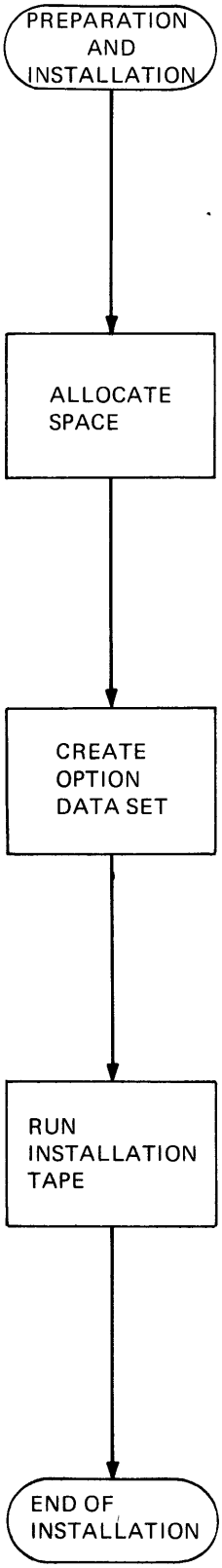
Preparation and installation of the compiler, transient library, and resident library, should be carried out according to the procedures described in this section. Figure 1 (compiler), figure 4 (transient library), and figure 6 (resident library) summarize the procedures.

Preparing for Compiler Installation

Figure 1 shows the three jobs which must be carried out before the compiler installation tape can be run. Each of these jobs, numbered 1, 2, and 3, corresponding to the sequence numbers in figure 1, are detailed below. Immediately following job 3, "Starting Reader To Tape", is an example of all the JCL required for the first two preparatory jobs.

(1) ALLOCATING SPACE FOR THE COMPILER

Before the compiler can be installed, space must be allocated for it in your operating system. The data sets needed must be allocated with sufficient space (see "Storage Estimates" section) and be cataloged. They are:



1. Allocate space for the compiler in a link library, IBM supplied cataloged procedures and LOGON procedure, if required, in a procedure library, and space for the TSO command processor in system subroutine libraries, if required.

2. Use the two macro instructions, PLIXOPB and PLIXOPC, to change the defaults for compile-time options, and also to delete any options not required. Create an option data set, SYS1.PPOPTION, and add your macro specifications using IEBUPDTE.

3. Start a reader to the installation tape.

Figure 1. Preparation and installation (compiler)

```
//ALLOC      EXEC  PGM=IEHLIST
//SYSPRINT   DD    SYSOUT=A
//LINK       DD    DSN=PLI.LINKLIB,UNIT=2311,DISP=(NEW,CATLG),
//                 VOL=(PRIVATE,RETAIN,SER=231100),
//                 SPACE=(CYL,(50,2,30))
//PROC       DD    DSN=PLI.PROCLIB,UNIT=2311,DISP=(NEW,CATLG),
//                 DCB=(DSORG=PO,RECFM=FB,BLKSIZE=80,LRECL=80),
//                 VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(1,1,2))
//CMDL       DD    DSN=PLI.CMDLIB,UNIT=2311,DISP=(NEW,CATLG),
//                 VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(2,2,5))
//HELP       DD    DSN=PLI.HELP,UNIT=2311,DISP=(NEW,CATLG),
//                 VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(1,1,2))
//SYSIN      DD    *
 LISTVTOC          VOL=2311=231100  * CHECK LIST DATA SET ALLOCATION *
/*
```

- SYS1.LINKLIB or private library — Required. Used for compiler modules.
- SYS1.PROCLIB or private library — Required. Used for IBM supplied batch mode cataloged procedures. Also used for IBM supplied compiler TSO LOGON procedure if requested.
- SYS1.CMDLIB or private library — Optional. Used for optimizing compiler TSO command processor (prompter) modules.
- SYS1.LINKLIB or private library — Required. Used for extended precision floating point simulator module.
- SYS1.HELP or private library — Optional. Used for optimizing compiler TSO HELP modules.

Note:

1. The optional data sets, for the TSO command processor and HELP modules are only used, and therefore need only be allocated, if the option relating to those data sets are requested in the optimizing compiler system generation macro specifications, (see "Creating the Option Data Set" below).
2. The choice of whether to use system or private libraries is discussed earlier under "Preparing for Installation (General)." If existing data sets are used, then the OS move/copy utility may be used to re-allocate them if sufficient space is not available, (see "Storage Estimates" section).

Sample JCL to allocate and catalog optimizing compiler data sets is shown above. The SPACE parameters used are for illustration purposes only; use the "Storage Estimates" section for actual values.

Before the compiler installation job stream can be run, you must add a cataloged procedure, PLIXDEF, to your system procedure library (normally SYS1.PROCLIB). This procedure is used by the job stream to access the data sets you allocate, and must contain DD statements with standard ddnames for all the data sets listed above. The procedure must also include a DD statement, SLNK, which must specify the dsname of the main system link library data set (normally SYS1.LINKLIB) containing member IEAXPALL. This is an extended precision floating point simulator module, and is included during the linkage edit process for compiler phases IEL0PAL2, 3, and 5. The IEHLIST utility may be used to determine the data set required if your system has a multiple data set system link library. The JCL needed to add PLIXDEF to SYS1.PROCLIB is as follows:

```
//DEFINE    EXEC    PGM=IEBUPDTE,PARM='NEW'
//SYSPRINT  DD      SYSOUT=A
//SYSUT2    DD      DSN=SYS1.PROCLIB,DISP=OLD
//SYSIN     DD      DATA
./          ADD     NAME=PLIXDEF,LIST=ALL
./          NUMBER  NEW=10,INCR=10
//PLIX      EXEC    PGM=IEHLIST
//SYSPRINT  DD      DUMMY
//SYSIN     DD      DUMMY
//LINK      DD      DSN=PLI.LINKLIB,DISP=OLD    * PRIVATE LINK LIBRARY *
//PROC      DD      DSN=PLI.PROCLIB,DISP=OLD    * PRIVATE PROC LIBRARY *
//CMDL      DD      DSN=PLI.CMDLIB,DISP=OLD     * PRIVATE CMD LIBRARY *
//HELP      DD      DSN=PLI.HELP,DISP=OLD       * PRIVATE HELP LIBRARY *
//SLNK      DD      DSN=SYS1.LINKLIB,DISP=SHR   * SYSTEM LINK LIBRARY *
/*
```

Note:

1. PLIXDEF is the procedure name
2. PLIX is the step name
3. LINK is the compiler link library ddname
4. PROC is the procedure library ddname
5. CMDL is the TSO command processor library ddname (If the TSO command processor is not required the statement should be coded as follows: //CMDL DD DUMMY. It may not be omitted or JCL errors will result when the installation job stream is run.)
6. HELP is the TSO HELP library ddname (If the TSO HELP module is not required the statement should be coded as follows: //HELP DD DUMMY. It may not be omitted or JCL errors will result when the installation job stream is run.)
7. SLNK is the system link library ddname.

These items in the above procedure must be specified exactly as shown.

(2) CREATING THE OPTION DATA SET

Two macro instructions, PLIXOPB and PLIXOPC, are provided enabling you to specify and delete default compile-time options. The PLIXOPB macro instruction is used for the batch processing mode, and PLIXOPC for the conversational processing mode (i.e., when compiler is invoked under TSO). The defaults for the compiler processing options are indicated in figure 3 and in the discussion of each option following the figure.

Note: If you are going to use only the IBM default compile-time options you must still specify the macro instructions PLIXOPB and PLIXOPC, but with no operands. For an example of PLIXOPC with no operands, refer to figure 2.

Before the installation job stream is run, you must first:

- Define and catalog a partitioned data set named SYS1.PPOPTION.
- Provide your specifications of PLIXOPB and PLIXOPC, containing the options you want, as members of SYS1.PPOPTION.

It is recommended that you allocate and create SYS1.PPOPTION at the same time you run the job steps to allocate space for the compiler.

The members PLIXOPB and PLIXOPC must contain the PLIXOPB (batch mode) and PLIXOPC (conversational mode) macro instructions respectively to describe the default values desired by your installation. The format of the PLIXOPB and PLIXOPC macro instructions is given under the heading "Default Compile-time Options."

When you run the installation tape, the PLIXOPB and PLIXOPC macro instructions are scanned for syntax errors and incorrectly specified keyword parameters. If an error is found, processing is terminated and a message describing the error is issued. The rest of the installation job stream is flushed. Correct the error in the PLIXOPB or PLIXOPC macro instruction and rerun the installation tape.

```
//OPTION    EXEC     PGM=IEBUPDTE,PARM='NEW'
//SYSPRINT  DD       SYSOUT=A
//SYSUT2    DD       DSN=SYS1.PPOPTION,UNIT=2311,VOL=(PRIVATE,RETAIN,SER=231100),
//                   DCB=(DSORG=PO,RECFM=FB,BLKSIZE=3440,LRECL=80),
//                   VOL=(PRIVATE,RETAIN,SER=231100),
//                   SPACE=(CYL,(1,1,1))DISP=(,CATLG)
//SYSIN     DD       *
./          ADD      NAME=PLIXOPB,LIST=ALL  * BATCH MODE OPTIONS *
            PLIXOPB  ATTRIBU=YES,ESD=YES,NEST=YES,LINECOU=50,
                     DELETE=(MDECK,NUMBER)
            END
./          ADD      NAME=PLIXOPC,LIST=ALL  * CONVERSATIONAL MODE OPTIONS *
            PLIXOPC
            END
/*
```

Example: This example illustrates the job control language statements needed to define and allocate the SYS1.PPOPTION data set and the members PLIXOPB and PLIXOPC. The example shows a sample use of the PLIXOPB and PLIXOPC macro instructions to tailor the optimizing compiler.

The macro specification for PLIXOPB requests that the batch mode default compile-time options will be modified to specify:

- Printing of an Attribute Table.
- Printing of the External Symbol Dictionary.
- Indicating for each statement, its begin block level and its DO-group level on the source program listing.
- Printing 50 lines on each page of the compiler output listing.
- Deleting the MDECK and NUMBER options

The macro specification for PLIXOPC requests that no LOGON procedure, or command processor will be generated, and that the IBM standard conversational mode default compile-time options will not be modified (this specification of PLIXOPC would normally be used, for example, in an MFT environment, which does not support TSO).

Installing Compiler

(3) STARTING READER TO TAPE

When you have run the job to allocate space for the compiler and create the option data set, you can start a reader to the IBM installation job stream tape.

When the installation tape is mounted, you ready the tape drive and enter a start reader command from the system console. For details of the start reader command, and of running the installation tape, refer to "Using Installation Tapes" to be found earlier in this section of the manual.

Example 1 - User Specified JCL for the Compiler

Figure 2 below shows an example of the JCL required to be run before using the installation job stream tape to install the compiler. The job steps illustrated perform the following functions:

Step DEFINE - Adds the cataloged procedure PLIXDEF defining the compiler link, system link, procedure, and subroutine library data sets, PLI.LINKLIB, SYS1.LINKLIB, PLI.PROCLIB, PLI.CMDLIB, and PLI.HELP to the system procedure library SYS1.PROCLIB.

Step CLEAN - Ensures that the target and system generation macro specification libraries have not already been wrongly allocated or cataloged. This step is not essential, but provides a safeguard in the event of previous faulty allocation.

Step ALLOC - This step does the following:

- Allocates the private link, system link, procedure, and subroutine library data sets PLI.LINKLIB, SYS1.LINKLIB, PLI.PROCLIB, PLI.CMDLIB, and PLI.HELP respectively, on the 2311 volume labeled 231100.
- Allocates the system generation macro specification library, SYS1.PPOPTION, on the same volume.
- Ensures, by using dummy DD statements, that sufficient work space exists on the SYS1.PPOPTION volume and other public volumes to run the compiler job stream.

- Catalogs the data sets PLI.LINKLIB, PLI.PROCLIB, PLI.CMDLIB, PLI.HELP, and SYS1.PPOPTION.

Step OPTION - Adds the two compiler system generation macro specifications, PLIXOPB and PLIXOPC, to SYS1.PPOPTION. PLIXOPB specifies that the non standared default compile time options: ATTRIBUTE, GOSTMT, OPTIMIZE(TIME) are to be generated as defaults for batch mode compiler use. PLIXOPC is specified with one operand PROMPT=YES specifying inclusion of the TSO command processor. Standard default compile-time options are to be generated for conversational mode processing. Note the assembler END statement required after each macro instruction.

After this job has been run, the compiler installation job stream itself is run using the following command to read the installation tape from unit 182:

```
START RDR,182,,,DCB=(RECFM=FB,
  BLKSIZE=3440,BUFL=3440,DEN=2)
```

During the job stream processing, the compiler modules will be added to PLI.LINKLIB; the cataloged procedures PLIXC, PLIXCL, PLIXCG, and PLIXCLG and the TSO LOGON procedure will be added to PLI.PROCLIB; the TSO command processor will be added to PLI.CMDLIB; the TSO HELP module will be added to PLI.HELP; and the compiler sample program for batch mode will be punched down on the device used for system output class B.

Finally, before the compiler can be used, PLI.LINKLIB should be concatenated to the standard system link library using the Link Library List option. PLI.PROCLIB may either be copied to SYS1.PROCLIB, or used in a special reader procedure. For more information on the Link Library List option and use of a special reader procedure refer to the publication: OS System Programmer's Guide.

```
//COMPILER JOB 1,CUSTOMER,MSGLEVEL=1
//*
//*********************************************************************
//*                                                                   ***
//* EXAMPLE OF PREPARATION STEPS TO BE DONE BEFORE RUNNING PL/I       ***
//* OPTIMIZING COMPILER INSTALLATION JOB STREAM.                      ***
//*                                                                   ***
//*********************************************************************
//*
//DEFINE EXEC  PGM=IEBUPDTE,PARM='NEW'
//*
//* ADD CATALOGED PROCEDURE 'PLIXDEF', DEFINING TARGET LIBRARIES TO
//* BE USED BY COMPILER INSTALLATION JOB STREAM, TO PROCEDURE LIBRARY.
//*
//SYSPRINT DD  SYSOUT=A
//SYSUT2   DD  DSN=SYS1.PROCLIB,DISP=OLD
//SYSIN    DD  DATA
./       ADD   NAME=PLIXDEF,LIST=ALL
./      NUMBER NEW1=10,INCR=10
//PLIX   EXEC  PGM=IEHLIST
//SYSPRINT DD  DUMMY
//SYSIN    DD  DUMMY
//LINK     DD  DSN=PLI.LINKLIB,DISP=OLD     * PRIVATE LINK LIBRARY *
//PROC     DD  DSN=PLI.PROCLIB,DISP=OLD     * PRIVATE PROC LIBRARY *
//CMDL     DD  DSN=PLI.CMDLIB,DISP=OLD      * PRIVATE CMD LIBRARY *
//HELP     DD  DSN=PLI.HELP,DISP=OLD        * PRIVATE HELP LIBRARY *
//SLNK     DD  DSN=SYS1.LINKLIB,DISP=SHR    * SYSTEM LINK LIBRARY *
/*
//CLEAN  EXEC  PGM=IEHPROGM
//*
//* ENSURE NEW TARGET LIBRARIES DO NOT ALREADY EXIST ON TARGET VOLUME.
//*
//SYSPRINT DD  SYSOUT=A
//TARGET   DD  UNIT=2311,VOL=(PRIVATE,RETAIN,SER=231100),DISP=OLD
//SYSIN    DD  *
```

Figure 2. (Part 1 of 2). Example of user specified JCL - compiler

```
 SCRATCH DSNAME=PLI.LINKLIB,VOL=2311=231100    * TARGET LINK LIBRARY *
 SCRATCH DSNAME=PLI.PROCLIB,VOL=2311=231100    * TARGET PROC LIBRARY *
 SCRATCH DSNAME=PLI.CMDLIB,VOL=2311=231100     * TARGET CMD LIBRARY *
 SCRATCH DSNAME=PLI.HELP,VOL=2311=231100       * TARGET HELP LIBRARY *
 SCRATCH DSNAME=SYS1.PPOPTION,VOL=2311=231100 * OPTIONS SPECIFICATION *
 UNCATLG DSNAME=PLI.LINKLIB
 UNCATLG DSNAME=PLI.PROCLIB
 UNCATLG DSNAME=PLI.CMDLIB
 UNCATLG DSNAME=PLI.HELP
 UNCATLG DSNAME=SYS1.PPOPTION
/*
//ALLOC EXEC  PGM=IEHLIST
//*
//*    1. ALLOCATE LINK, PROCEDURE AND SUBROUTINE TARGET LIBRARIES.
//*
//*    2. ALLOCATE OPTIONS MACRO SPECIFICATION LIBRARY.
//*
//*    3. ENSURE SUFFICIENT WORK SPACE EXISTS ON TARGET VOLUME AND OTHER
//*       NON-PRIVATE VOLUMES TO RUN INSTALLATION JOB STREAM.
//*
//*    4. CATALOG LINK, PROCEDURE, SUBROUTINE, & OPTIONS SPECIFICATION LIBRARIES.
//*
//SYSPRINT DD  SYSOUT=A
//LINK     DD  DSN=PLI.LINKLIB,UNIT=2311,DISP=(NEW,CATLG),
//             VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(50,2,30))
//PROC     DD  DSN=PLI.PROCLIB,UNIT=2311,DISP=(NEW,CATLG),
//             DCB=(DSORG=PO,RECFM=FB,BLKSIZE=80,LRECL=80),
//             VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(1,1,2))
//CMDL     DD  DSN=PLI.CMDLIB,UNIT=2311,DISP=(NEW,CATLG),
//             VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(2,2,5))
//HELP     DD  DSN=PLI.HELP,UNIT=2311,DISP=(NEW,CATLG),
//             VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(1,1,2))
//OPTIONS  DD  DSN=SYS1.PPOPTION,UNIT=2311,DISP=(NEW,CATLG),
//             DCB=(DSORG=PO,RECFM=FB,BLKSIZE=3440,LRECL=80),
//             VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(1,1,1))
//*
//* THE FOLLOWING DD STATEMENTS CHECK THAT THERE IS SUFFICIENT BASIC
//* WORK SPACE ON THE TARGET AND OTHER VOLUMES TO RUN THE INSTALLATION
//* JOB STREAM, THEY ARE DELETED AT THE END OF THE STEP.
//*
//WORK1    DD  UNIT=SYSSQ,SPACE=(3072,40)
//WORK2    DD  UNIT=SYSSQ,SPACE=(3072,40)
//WORK3    DD  UNIT=SYSDA,SPACE=(3072,60)
//WORK4    DD  UNIT=SYSDA,SPACE=(80,(120,,1))
//WORK5    DD  UNIT=2311,VOL=(PRIVATE,RETAIN,SER=231100),
//             SPACE=(CYL,4)
//SYSIN    DD  *
 LISTVTOC VOL=2311=231100      * CHECK LIST DATA SET ALLOCATION *
/*
//OPTION EXEC  PGM=IEBUPDTE,PARM='NEW'
//*
//* ADD MACRO DEFINITIONS DEFINING COMPILER OPTIONS REQUIRED TO
//* SYS1.PPOPTION DATA SET ALLOCATED IN PREVIOUS STEP.
//*
//SYSPRINT DD  SYSOUT=A
//SYSUT2   DD  DSN=SYS1.PPOPTION,VOL=(PRIVATE,RETAIN),DISP=OLD
//SYSIN    DD  *
./       ADD   NAME=PLIXOPB,LIST=ALL   * BATCH MODE OPTIONS.
     PLIXOPB   ATTRIBU=YES,GOSTMT=YES,OPTIMIZ=TIME
     END
./       ADD   NAME=PLIXOPC,LIST=ALL    * CONVERSATIONAL MODE OPTIONS.
     PLIXOPC
     END
/*
//
```

Figure 2. (Part 2 of 2). Example of user specified JCL - compiler

SAMPLE PROGRAM

The sample program is punched down on cards, by IEBGENER, as the final job on the compiler installation tape. You can run this program both to check satisfactory installation, and also to illustrate use of the compiler. This sample program can subsequently be run to check satisfactory installation of the transient and resident libraries.

The sample program illustrates the use of the compile-time preprocessor to convert source programs written for the PL/I(F) Compiler for compilation by the OS PL/I Optimizing Compiler.

Note: You will be unable to linkage edit and execute the sample program before installing the transient and resident libraries. Under normal circumstances you will install the compiler and the libraries before running the sample program.

Sample Program Output

For sample program output refer to the publication: OS PL/I Optimizing Compiler: Programmer's Guide.

DEFAULT COMPILE-TIME OPTIONS

Figure 3 shows the compile-time options; the operands underscored are the standard defaults. A full description of every option follows the figure.

The following conventions are used to illustrate the format and coding of compile-time options in figure 3.

- Upper case letters, numbers, and punctuation marks must be coded by the programmer exactly as shown. Exceptions to this convention are brackets, [ ] and braces, { }. These are never coded.
- Lower case letters and words represent variables for which the programmer must substitute specific information or specific values.
- Items or groups of items within brackets, [ ] are optional. They may be omitted at the programmer's discretion. Conversely, the lack of brackets indicates that an item or group of items must be coded.
- Braces, { }, group related items.
- Stacked items enclosed in braces represent alternative items. Only one of the stacked items should be coded. The 'Or' sign is also used in figure 3 to denote this.

Name	Operation	Operand	Defaults	
			PLIXOPB	PLIXOPC
	PLIXOPB & PLIXOPC	*[PROMPT= {YES / NO}]	NO	NO
		[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
		['charstring']	OPTIMIZE	OPTIMIZE
		[DECK= {YES / NO}]	NO	NO
		[ESD= {YES / NO}]	NO	NO
		[FLAG= {I / W / E / S}]	I	W
		[FLOW= {(n,p) / NO}]	NO	NO
		[GONUMBE= {YES / NO}]	NO	NO
		[GOSTMT= {YES / NO}]	NO	NO
		[IMPRECI= {YES / NO}]	NO	NO
		[INSOURC= {YES / NO}]	YES	NO

Figure 3. (Part 1 of 3). Default compile-time options

Name	Operation	Operand	Defaults	
			PLIXOPB	PLIXOPC
	PLIXOPB & PLIXOPC (continued)	[LINCOU= 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
		[MARGINS= ({m},{n}[,c])]	2,72	2,72
		[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

Figure 3. (Part 2 of 3). Default compile-time options

Name	Operation	Operand	Defaults	
			PLIXOPB	PLIXOPC
	PLIXOPB & PLIXOPC (continued)	[TERMINA= {YES / NO / (option list)}]	NO	YES
		[XREF= {NO / YES}]	NO	NO
		*[DELETE= (item[, item[,...]])]	-	-

Operand Field: All the parameters of the PLIXOPB and PLIXOPC system generation macros specify the setting of default options at compilation time. Default options are the options that are assumed if the programmer does not supply alternative values at compile-time via the JCL EXEC statement, or the PL/I PROCESS statement. Default values will always be used at compile-time for any options specified in the DELETE operand.

* Denotes operands that can only be used at the time of installing the compiler.

Figure 3. (Part 3 of 3). Default compile-time options

The following is a detailed description of the operands:

PROMPT=

a conversational aids operand, which specifies whether or not the optimizing compiler TSO command processor (prompter), and LOGON cataloged procedure are to be generated.

YES

specifies that the command processor and LOGON cataloged procedure will be generated.

NO

specifies that the command processor and LOGON cataloged procedure will not be generated.

Default: If this keyword parameter is omitted, NO is assumed.

AGGREGA=

specifies the production of an 'AGGREGATE LENGTH TABLE', giving the length in bytes of all major structures and non-structured arrays in the PL/I program.

YES

specifies that a table is to be produced.

NO

specifies that a table is not to be produced.

Default: If this keyword parameter is omitted, NO is assumed.

ATTRIBU=

specifies the production of an 'ATTRIBUTE TABLE' that lists all the identifiers used in the PL/I program together with their attributes

YES

specifies that a table is to be produced.

NO

specifies that a table is not to be produced.

Default: If this keyword parameter is omitted, NO is assumed.

CHARSET=

specifies the character code used to represent your program and the character set in which you have written the program.

EBCDIC

specifies that the character code used is to be EBCDIC.

BCD

specifies that the character code used is to be BCD.

48

specifies that the 48 character set is to be used.

60

specifies that the 60 character set is to be used.

Default: If this keyword parameter is omitted, EBCDIC and 60 are assumed.

COMPILE=

specifies whether the compiler is to compile the source program if errors occurred during syntax checking.

YES

specifies that the source program is to be compiled unconditionally.

NO

specifies that the source program is not to be compiled.

NOW

specifies that compilation is to be suppressed when any warning, error, severe, or unrecoverable errors are detected.

NOE

specifies that compilation is to be suppressed when any error, severe, or unrecoverable errors are detected.

NOS

specifies that compilation is to be suppressed when any severe or unrecoverable errors are detected.

Default: If this keyword parameter is omitted, NOS is assumed.

CONTROL=

specifies a character string which must be used as a password before an option, included in the DELETE operand, can be respecified at compile-time.

'charstring'

specifies a name which may be used to enable compiler options deleted at system generation to be used for a particular compilation. The password must be 1 to 8 characters. If less than 8 characters, the value will be right padded with the equivalent of hexadecimal zeros up to 8 bytes.

Note: This operand is provided to enable system programmers to use 'deleted' options to obtain additional information (e.g. LIST, if normally deleted) in the event of programming problems. Therefore, it is advisable to restrict knowledge of the 'charstring' to such personnel.

Default: If this keyword parameter is omitted, OPTIMIZE is assumed.

DECK=

specifies that the compiler is to write the object module in the form of 80-column card images onto SYSPUNCH.

YES

specifies that the object module is to be written onto SYSPUNCH.

NO

specifies that the object module is not to be written onto SYSPUNCH.

Default: If this keyword parameter is omitted, NO is assumed.

ESD=

specifies the inclusion of a listing of the external symbol dictionary.

YES

specifies that a listing of the external symbol dictionary is to be included.

NO

specifies that a listing of the external symbol dictionary is not to be included.

Default: If this keyword parameter is omitted, NO is assumed.

FLAG=

specifies the minimum level of severity that requires a message to be printed.

I

specifies that all diagnostic messages are to be listed.

W

specifies that all diagnostic messages, except "informatory" messages are to be listed.

E

specifies that all diagnostic messages, except "warning" and "informatory" messages are to be listed.

S

specifies that only "severe", and "unrecoverable" errors are to be listed.

Default: If this keyword parameter is omitted, I is assumed for batch mode, and W is assumed for conversational mode.

FLOW=

specifies that the compiler is to insert code that will give the numbers of the last "n" source statements executed prior to the occurrence of an interrupt that results in an execution-time diagnostic message.

(n,p)

specifies that the compiler is to give the number of the last "n" source statements executed prior to the occurrence of an interrupt. The number of statement numbers is specified by "n". The maximum permitted for "n" is 32,768. The second argument "p" of this option defines the number of procedures through which a flow-trace is to be maintained at any time.

NO

specifies that numbers of source statements executed prior to the occurrence of an interrupt will not be given by the compiler.

Default: If this keyword parameter is omitted. NO is assumed.

GONUMBE=

specifies inclusion of source program line numbers in execution-time messages.

YES

specifies that execution-time messages will contain source program line numbers.

NO

specifies that execution-time messages will not contain source program line numbers.

Default: If this keyword parameter is omitted, NO is assumed.

GOSTMT

specifies additional information that will allow statement numbers from the source program to be included in diagnostic messages produced during execution of the compiled program.

YES

specifies that the compiler is to produce additional information

NO

specifies that the compiler is not to produce additional information.

Default: If this keyword parameter is omitted, NO is assumed.

IMPRECI=

specifies that the compiler is to generate additional machine instructions that will process imprecise interrupts when the compiled program is executed on a Model 91 or 195.

YES

specifies that the compiler is to generate the additional machine instructions.

NO

specifies that the compiler is not to generate the additional machine instructions.

Default: If this keyword parameter is omitted, NO is assumed.

INSOURC=

specifies a listing of the PL/I source statements by the preprocessor.

YES

specifies a listing is produced.

NO

specifies a listing is not produced.

Default: If this keyword parameter is omitted, YES is assumed for batch mode, and NO is assumed to conversational mode.

LINECOU=

specifies the number of lines to be included in each page of a printed listing, including heading lines and blank lines.

n

is an integer from 10 to 32,767.

Default: If this keyword parameter is omitted, 55 is assumed.

LIST=

specifies a listing of the object module generated by the compiler.

YES

specifies that a listing is to be produced.

NO

specifies that a listing is not to be produced.

Default: If this keyword parameter is omitted, NO is assumed.

LMESSAG=

specifies the length of messages produced by the compiler.

YES

specifies that the messages produced by the compiler are to be in the long form.

NO

specifies that the messages produced by the compiler are to be in the short form.

Default: If this keyword parameter is omitted, YES is assumed for batch mode, and NO is assumed for conversational mode.

MACRO=

specifies the employment of the compile-time preprocessor.

YES

specifies that you want to employ the compile-time preprocessor.

NO

specifies that you do not want to employ the compile-time preprocessor.

Default: If this keyword parameter is omitted, NO is assumed.

MAP=

specifies the printing of tables showing the organization of the static storage for the compiled object module. These tables consist of a static internal storage map and the static external control sections.

YES

specifies that the tables should be printed.

NO

specifies that the tables should not be printed.

Default: If this keyword parameter is omitted, NO is assumed.

MARGINI=

specifies that the compiler is to print the specified character on the margins of the source listing, thus revealing any source statements that cross either margin.

'character'

specifies the character that the compiler is to print on the margins of the source listing.

NO

specifies that no character is to be printed in the margins of the source listing.

Default: If this keyword parameter is omitted, NO is assumed.

MARGINS=

specifies the extent of the part of each input record that contains the PL/I source statements. The compiler will not process data that is outside these limits.

(m,n[,c])

specifies that 'm' represents the position in the input record of the first byte of the field that contains the source statement record, that 'n' represents the position in the input record of the last byte of the source statement field, and that 'c' represents the position in the input record of the byte that will contain the control character. The value 'm' must be less than or equal to 'n', and neither must exceed 100. The value 'c' must be outside the limits set by 'm' and 'n' and must not exceed 100.

Default: If this keyword parameter is omitted, 2,72 is assumed.

MDECK=

specifies that output from the preprocessor should be in the form of a card deck. This output is written (punched) as a data set on SYSPUNCH.

YES

specifies that you want the preprocessor output in the form of a card deck.

NO

specifies that you do not want the preprocessor output in the form of a card deck.

Default: If this keyword parameter is omitted, NO is assumed.

NEST=

specifies that for each statement, its begin-block level and its DO-group level should be indicated on the source program listing.

YES

specifies that the source program listing should indicate for each statement, its begin-block and DO-group level.

NO

specifies no indication of begin-block and DO-group levels for statements.

Default: If this keyword parameter is omitted, NO is assumed.

NUMBER=

specifies that the source program contains line numbers.

YES

specifies inclusion of source program line numbers.

NO

specifies that source program line numbers are not to be included.

Note: The STMT option is implied if NONUMBER is specified at compilation time.

Default: If this keyword parameter is omitted, NO is assumed for batch mode, and YES is assumed for conversational mode.

OBJECT=

specifies that the compiler is to produce an object module in a form suitable for input to the linkage editor.

YES

specifies that an object module should be produced.

NO

specifies that an object module should not be produced.

Default: If this keyword parameter is omitted, YES is assumed.

OFFSET=

specifies that execution time diagnostic messages will include offset addresses relative to the primary entry point of the procedure.

YES

specifies that offsets should be printed.

NO

specifies that offsets should not be printed.

Default: If the keyword parameter is omitted, NO is assumed.

OPTIMIZ=

specifies the type of optimization required.

NO or 0

specifies fastest possible compilation, but inhibits optimization for faster execution speeds and reduced main storage requirements.

TIME or 2

specifies that the compiler is to optimize the machine instructions generated for minimum execution time. A secondary effect of this type of optimization can be a reduction in the amount of main storage required for the object module. The use of OPTIMIZE (TIME) could result in a substantial increase in compile-time over NOOPTIMIZE.

Default: If the keyword parameter is omitted, NO or 0 is assumed.

OPTIONS=

specifies a list showing the status of all the compiler options after any default attributes have been applied at the start of compilation.

YES

specifies that a list should be produced.

NO

specifies that a list should not be produced.

Default: If the keyword parameter is omitted, YES is assumed for batch mode, and NO is assumed for conversational mode.

SIZE=

specifies the amount of main storage available for the compilation. The amount is limited to the size of the region available to the compiler for MVT systems, or the size of the partition available to the compiler for MFT systems.

n

specifies that yyyyyy bytes of main storage are available for the compilation. Leading zeros are not required.

nk

specifies that yyyK bytes of main storage are available for the compilation. Leading zeros are not required.

MAX

specifies that the compiler is to obtain as much main storage as it can.

Default: If the keyword parameter is omitted, MAX is assumed.

SOURCE=

specifies a listing of the PL/I source statements processed by the compiler. The source statements listed are either those of the original source program or the output from the preprocessor.

YES

specifies that a listing should be produced.

NO

specifies that a listing should not be produced.

Default: If the keyword parameter is omitted, YES is assumed for batch mode, and NO is assumed for conversational mode.

STMT=

specifies the inclusion of statement numbers in the source listing.

YES

specifies that statement numbers are to be included.

NO

specifies that statement numbers are not to be included.

Note: The NUMBER option is implied if NOSTMT is specified at compilation time.

Default: If the keyword parameter is omitted, YES is assumed for batch mode, and NO is assumed for conversational mode.

STORAGE=

specifies the printing of a table giving the storage requirements for the compiled object-module.

YES

specifies that the table is to be printed.

NO

specifies that the table is not to be printed.

Default: If the keyword parameter is omitted, NO is assumed.

SYNTAX=

specifies syntax checking depending on the severity of errors detected by the preprocessor.

YES

specifies that syntax checking should continue unconditionally.

NO

specifies that syntax checking should not continue.

NOW

specifies that syntax checking should be suppressed if an unrecoverable error, severe error, error, or warning is detected by the preprocessor.

NOE

specifies that syntax checking should be suppressed if an unrecoverable error, severe error, or an error is detected by the preprocessor.

NOS

specifies that syntax checking should be suppressed if an unrecoverable error, or a severe error is detected by the preprocessor.

Default: If the keyword parameter is omitted, NOS is assumed.

TERMINA=

specifies that diagnostic and informational output is to be printed at the terminal.

YES

specifies that the information is to be printed at the terminal.

NO

specifies that the information is not to be printed at the terminal.

(option list)

specifies that in addition to printing diagnostic and informational output at the terminal, the listings given on the 'option list' are also to be printed at the terminal.

The following options, their negative forms, or their abbreviated forms, can be specified in the option list:

AGGREGATE	OFFSET
ATTRIBUTES	OPTIONS
ESD	SOURCE
INSOURCE	STORAGE
LIST	XREF
MAP	

Default: If the keyword parameter is omitted, YES is assumed for conversational mode, and NO is assumed for batch mode.

XREF=

specifies the printing of a cross-reference table that lists all the identifiers in the source program with the numbers of the source statements in which they appear.

YES

specifies that the cross-reference table is to be printed.

NO

specifies that the cross-reference table is not to be printed.

Default: If the keyword parameter is omitted, NO is assumed.

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

specifies that the values in the item list cannot be used at compilation time in the *PROCESS card or PARM field of the EXEC statement to override the

default options established by the PLIXOPB and PLIXOPC macros, unless the CONTROL option is also specified with the correct password. One or more of the following values can be specified; each has been described above.

AGGREGA	LINECOU	SOURCE
ATTRIBU	MAP	SYNTAX
CHARSET	MARGINI	STORAGE
LIST	MARGINS	MDECK
COMPILE	GONUMBE	IMPRECI
DECK	MACRO	LMESSAG
ESD	NEST	NUMBER
FLAG	OFFSET	OBJECT
FLOW	OPTIMIZ	STMT
GOSTMT	OPTIONS	XREF
INSOURC	SIZE	TERMINA

Preparing for Transient Library Installation

Figure 4 shows the two jobs which must be carried out before the transient library installation tape can be run. Both of these jobs, numbered 1 and 2, corresponding to the sequence numbers in figure 4, are detailed below. Immediately following job 2, "Starting Reader To Tape", is an example of all the JCL required for the first of the two preparatory jobs.

(1) ALLOCATING SPACE FOR THE TRANSIENT LIBRARY

Before the transient library can be added to your operating system, space in a link library must be allocated for it.

All transient library modules must reside in a link library data set, which may be the same system link library data set SYS1.LINKLIB, or a private library. Note that if a private library is to be used you must specify a STEPLIB DD statement in your GO step to identify the transient library. Choice of library to be used is discussed earlier under "Preparing for Installation (General)." The procedure for allocating space for the transient library is the same as for the compiler, details of which will be found under the heading "Allocating Space for the Compiler."

A cataloged procedure named PLTRDEF, referencing the target link library data set, must be added to the system procedure library SYS1.PROCLIB, (see figure 5). The installation job stream uses the procedure to access the data set. The stepname (PLTR) and the link library ddname (LINK) must be coded as shown in figure 5. The LINK DD statement must not allocate space for the data set, as the procedure may be used more than once by the installation job stream.

Space for the link library data set can be allocated in the same way as that described for the compiler.

Installing Transient Library

(2) STARTING READER TO TAPE

When you have run the job to allocate space for the transient library, you can start a reader to the IBM installation job stream tape.

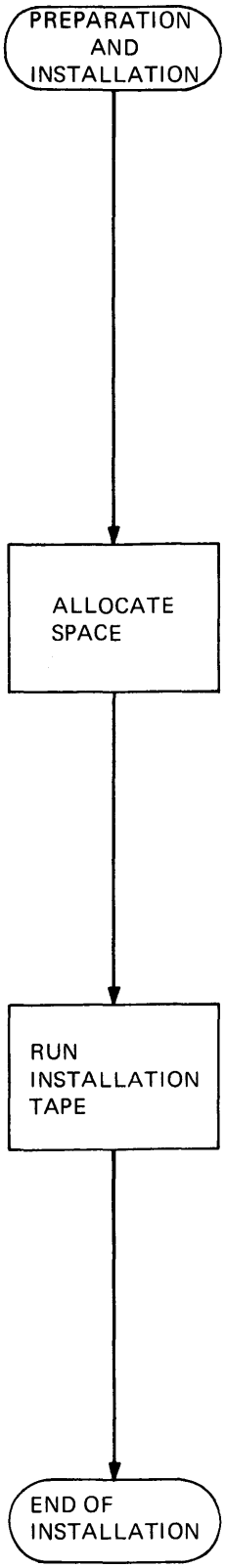
When the installation tape is mounted, you ready the tape drive and enter a start reader command from the system console. For details of the start reader command, and of running the installation tape, refer to "Using Installation Tapes" to be found earlier in this section of the manual.

Example 2 - User Specified JCL for the Transient Library

This example assumes that the compiler has already been installed as described in example 1, and uses the same link library data set, PLI.LINKLIB, to contain the transient library modules. Figure 5 below shows the JCL necessary to be run before using the installation job stream tape to install the transient library. PLI.LINKLIB is assumed to have been allocated with sufficient space for both the compiler and transient library (see "Storage Estimates" section). The job adds the cataloged procedure PLTRDEF defining PLI.LINKLIB as the target link library data set to the system procedure library SYS1.PROCLIB. No data set allocation or system generation macro specification steps are necessary.

Note: The procedure, step, and DD names in figure 5 must be coded as shown.

After running this job, the transient library installation tape is processed as described for the compiler in example 1, causing the transient library modules to be added to PLI.LINKLIB. Note that there are no cataloged procedures or sample program for this product.



① Allocate space for the transient library in a link library. Prepare a cataloged procedure named PLTRDEF referencing the link library data set, and add it to the procedure library SYS1.PROCLIB.

② Start a reader to the installation tape.

Figure 4. Preparation and installation (transient library)

```
//TRANSLIB JOB 1,CUSTOMER,MSGLEVEL=1
//*
//*********************************************************************
//*                                                                   ***
//* EXAMPLE OF PREPARATION STEPS TO BE DONE BEFORE RUNNING PL/I       ***
//* TRANSIENT LIBRARY INSTALLATION JOB STREAM.                        ***
//*                                                                   ***
//*********************************************************************
//*
//DEFINE EXEC  PGM=IEBUPDTE,PARM='NEW'
//*
//* ADD CATALOGED PROCEDURE 'PLTRDEF', DEFINING TARGET LINK LIBRARY
//* DATA SET TO BE USED BY TRANSIENT LIBRARY JOB STREAM, TO CATALOGED
//* PROCEDURE LIBRARY.
//*
//SYSPRINT DD  SYSOUT=A
//SYSUT2   DD  DSN=SYS1.PROCLIB,DISP=OLD
//SYSIN    DD  DATA
./       ADD   NAME=PLTRDEF,LIST=ALL
./      NUMBER NEW1=10,INCR=10
//PLTR   EXEC  PGM=IEHLIST
//SYSPRINT DD  DUMMY
//SYSIN    DD  DUMMY
//LINK     DD  DSN=PLI.LINKLIB,DISP=OLD
/*
//*
//* THIS EXAMPLE USES THE SAME TARGET LINK LIBRARY DATA SET,
//* PLI.LINKLIB, AS WAS USED TO INSTALL THE COMPILER PREVIOUSLY.
//* NO RE-ALLOCATION OR RE-CATALOGING OF IT IS REQUIRED.
//
```

Figure 5. Example of user specified JCL - transient library

Preparing for Resident Library Installation

Figure 6 shows the three jobs which must be carried out before the resident library installation tape can be run. Each of these jobs, numbered 1, 2, and 3, corresponding to the sequence numbers in figure 6, are detailed below. Immediately following job 3, "Starting Reader To Tape", is an example of all the JCL required for the first two preparatory jobs.

(1) ALLOCATING SPACE FOR THE RESIDENT LIBRARY

Before the resident library can be added to your operating system, space must be allocated for it, and a procedure named PLRSDEF, referencing the data sets (see Example 3), must be added to the system procedure library SYS1.PROCLIB. Space for the following libraries must be allocated:

- One or two subroutine libraries (the first is the basic resident library subroutine library, the second is for optional tasking modules).
- A procedure library for IBM supplied cataloged procedures.
- A private macro library for the shared library system generation macro.

Space for the resident library can be allocated in one of two ways, as described for the compiler. See also figure 8 which gives a complete example of all the preparatory jobs.

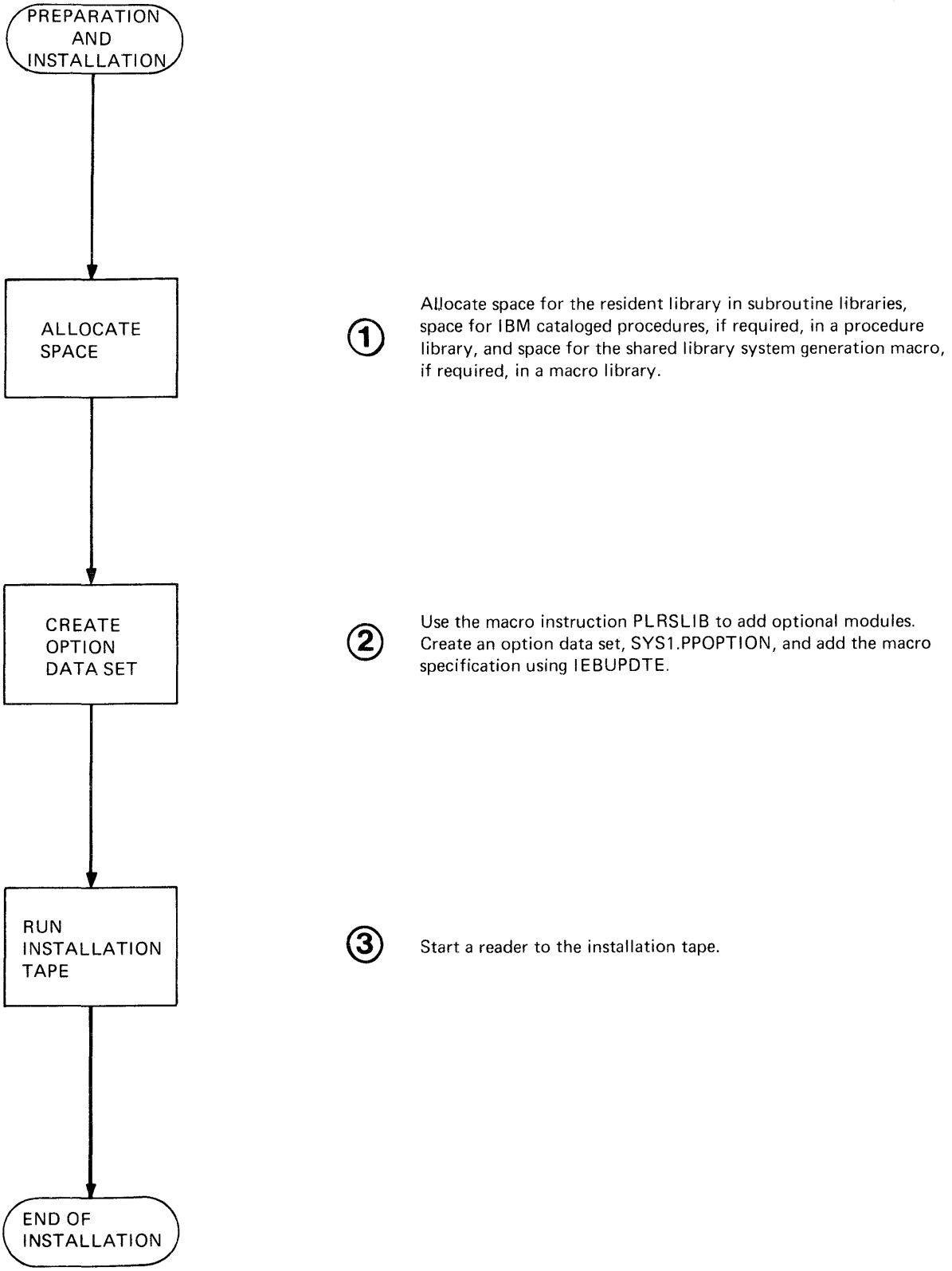


Figure 6. Preparation and installation (resident library)

(2) CREATING THE OPTION DATA SET

A macro instruction, PLRSLIB is provided to enable you to request the inclusion of complex functions, interlanguage communication, extended precision floating point subroutines, tasking modules, and the system generation macro PLRSHR.

Before the installation job stream is run, you must first:

- Define and catalog a partitioned data set named SYS1.PPOPTION (just as for the compiler).
- Specify the PLRSLIB macro defining the resident library options required, and add it, with an assembler END statement, to SYS1.PPOPTION.

It is recommended, as for the compiler, that you create the SYS1.PPOPTION data set at the same time you run the procedure for allocating space for the resident library.

The member PLRSLIB must contain the PLRSLIB macro instruction, even if your installation does not wish to use the optional subroutines.

Resident Library Options

The resident library options are shown in Figure 7.

Name	Operation	Operand
	PLRSLIB	[PLIBASE= option (option[,option[, option]])]
		[PLITASK= {YES / NO}]
		[PLRSHR= {YES / NO}]

Figure 7. Resident library options

The following is a detailed description of the operands:

PLIBASE=

The basic subroutine library data set, normally SYS1.PLIBASE, will always contain non-optional modules. This operand specifies additional optional groups of modules which may be added to it. Possible values for the 'option' sub-parameters are:

COMPLEX - modules to deal with complex arithmetic.

ILC - modules to deal with interlanguage communication.

EXTND - modules to deal with extended float arithmetic.

PLITASK={YES / NO}

specifies whether the data set SYS1.PLITASK is to be created or not. SYS1.PLITASK is the data set which has to be concatenated with SYS1.PLIBASE in order to run tasking programs.

PLRSHR={YES / NO}

specifies whether the shared library system generation macro (PLRSHR) is to be added to a private macro library for use later if the shared library is to be generated.

Note:
- If you do not want any optional groups of modules (both basic and tasking) or a shared library macro, code PLRSLIB without any operands.
- If you do not require tasking modules, then the TASKLIB DD statement in the PLRSDEF procedure (see Example 3) may be coded as:

```
//TASKLIB DD DUMMY
```

- If you do not require the PLRSHR macro then the SHRMAC DD statement in the PLRSDEF procedure may be coded as:

```
//SHRLIB DD DUMMY
```

TASKING and SHRMAC DD statements may not be omitted from the procedure, or JCL errors will result when the installation job stream is run.

Installing Resident Library

(3) STARTING READER TO TAPE

When you have run the job steps to allocate space for the resident library data sets, added PLRSDEF to SYS1.PROCLIB, and created SYS1.PPOPTION containing your PLRSLIB macro specification, you can start a reader to the IBM installation job stream tape.

When the installation tape is mounted, you ready the tape drive and enter a start reader command from the system console. For details of the start reader command, and of running the installation tape, refer to "Using Installation Tapes" to be found earlier in this section of the manual.

Example 3 - User Specified JCL for the Resident Library

The example assumes that the compiler and transient library have been installed previously as described in examples 1 and 2, and uses the same private procedure library PLI.PROCLIB, for the resident library cataloged procedures, and the same SYS1.PPOPTION data set. Figure 8 shows the JCL necessary to be run before using the installation tape to install the resident library. The job steps illustrated perform the following functions:

Step DEFINE - Adds the cataloged procedure PLRSDEF defining the procedure library, basic and tasking subroutine library data sets, and a link library if the shared library feature is to be used, (PLI.PROCLIB, SYS1.PLIBASE, SYS1.PLITASK, and SYS1.LINKLIB respectively) to the system procedure library SYS1.PROCLIB.

If you are not going to generate shared library modules after installing the resident library, the SHRLINK DD statement may be omitted from the procedure, but that for SHRMAC must be supplied but may be coded as described above.

Note: The procedure, step, and DD names in figure 8 must be coded as shown.

Step CLEAN - Ensures that SYS1.PLIBASE and SYS1.PLITASK have not already been wrongly allocated or cataloged. As in the compiler example, this step is not essential.

Step ALLOC - This step does the following:

- Allocates SYS1.PLIBASE and SYS1.PLITASK on the 2311 volume 231100 (the same pack as that used for the compiler and transient library). Note that PLI.PROCLIB and SYS1.PPOPTION were allocated when the compiler was installed (see Example 1).

- As the additional data sets SYS1.PLIBASE and SYS1.PLITASK have been allocated to the same volume as SYS1.PPOPTION, use a dummy DD statement to ensure that there is still sufficient work space on it to run the installation tape (other work space was checked when the compiler was installed, see Example 1).

- Catalogs SYS1.PLIBASE and SYS1.PLITASK.

Step OPTION - Adds the resident library system generation macro specification PLRSLIB to SYS1.PPOPTION. This macro specifies that complex functions, interlanguage communication, and extended precision floating point subroutines are to be included in SYS1.PLIBASE, and that tasking modules are to be added to SYS1.PLITASK. Also the shared library system generation macro is to be added to PLI.SHRMAC for later use. Note the assembler END statement following the macro specification.

After running the job, the resident library installation tape is processed in the same way as those for the compiler and transient library (see Example 1). During the job stream processing, the resident library modules will be added to SYS1.PLITASK and SYS1.PLIBASE; the IBM supplied cataloged procedures PLIXLG and PLIXG will be added to PLI.PROCLIB; and the resident library shared library system generation macro will be added to PLI.SHRMAC. The system generation macro can be used to generate the shared library feature after the resident library has been installed (see "Creating the Shared Library").

No further action is necessary to use the resident library, except to use the IEHPROGM utility to scratch SYS1.PPOPTION if it is no longer required.

```
//RESLIB   JOB1,CUSTOMER,MSGLEVEL=1
//*
//*********************************************************************
//* EXAMPLE OF PREPARATION STEPS TO BE DONE BEFORE RUNNING PL/I     ***
//* RESIDENT LIBRARY JOB STREAM.                                    ***
//*                                                                 ***
//*********************************************************************
//*
//DEFINE EXEC  PGM=IEBUPDTE,PARM='NEW'
//*
//* ADD CATALOGED PROCEDURE 'PLRSDEF', DEFINING TARGET LIBRARIES TO BE
//* USED BY RESIDENT LIBRARY INSTALLATION JOB STREAM, TO SYSTEM
//* PROCEDURE LIBRARY.
//*
//SYSPRINT DD  SYSOUT=A
//SYSUT2   DD  DSN=SYS1.PROCLIB,DISP=OLD
//SYSIN    DD  DATA
./       ADD   NAME=PLRSDEF,LIST=ALL
./      NUMBER NEW1=10,INCR=10
//PLRS   EXEC  PGM=IEHLIST
//SYSPRINT DD  DUMMY
//SYSIN    DD  DUMMY
//PROC     DD  DSN=PLI.PROCLIB,DISP=OLD     * PRIVATE PROC LIBRARY *
//BASELIB  DD  DSN=SYS1.PLIBASE,DISP=OLD    * BASIC SUBROUTINES    *
//TASKLIB  DD  DSN=SYS1.PLITASK,DISP=OLD    * TASKING SUBROUTINES  *
//SHRLINK  DD  DSN=SYS1.LINKLIB,DISP=OLD    * SHARED LIBRARY FEATURE *
//SHRMAC   DD  DSN=PLI.SHRMAC,DISP=OLD      * PRIVATE MACRO LIBRARY *
/*
//CLEAN  EXEC  PGM=IEHPROGM
//*
//* ENSURE NEW TARGET LIBRARIES DO NOT ALREADY EXIST ON TARGET VOLUME.
//*
//SYSPRINT DD  SYSOUT=A
//TARGET   DD  UNIT=2311,VOL=(PRIVATE,RETAIN,SER=231100),DISP=OLD
//SYSIN    DD  *
 SCRATCH DSNAME=SYS1.PLIBASE,VOL=2311=231100 * BASIC SUBROUTINE LIBRARY *
 SCRATCH DSNAME=SYS1.PLITASK,VOL=2311=231100 * TASKING LIBRARY *
 SCRATCH DSNAME=PLI.SHRMAC,VOL=2311=231100    * MACRO LIBRARY *
 UNCATLG DSNAME=SYS1.PLIBASE
 UNCATLG DSNAME=SYS1.PLITASK
 UNCATLG DSNAME=PLI.SHRMAC
/*
//ALLOC  EXEC  PGM=IEHLIST
//*
//*    1. ALLOCATE BASIC AND TASKING TARGET LIBRARIES
//*       AND PRIVATE MACRO LIBRARY.
//*
//*
//*    2. CATALOG BASIC AND TASKING SUBROUTINE LIBRARIES
//*       AND PRIVATE MACRO LIBRARY.
//*
```

Figure 8. (Part 1 of 2). Example of user specified JCL - resident library

```
//SYSPRINT DD  SYSOUT=A
//BASE     DD  DSN=SYS1.PLIBASE,UNIT=2311,DISP=(NEW,CATLG),
//             VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(10,2,55))
//TASK     DD  DSN=SYS1.PLITASK,UNIT=2311,DISP=(NEW,CATLG),
//             VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(5,2,20))
//SHRMAC   DD  DSN=PLI.SHRMAC,UNIT=2311,DISP=(NEW,CATLG),
//             VOL=(PRIVATE,RETAIN,SER=231100),SPACE=(CYL,(12,2,1)),
//             DCB=(DSORG=PO,RECFM=FB,BLKSIZE=3440,LRECL=80)
//*
//* THE FOLLOWING DD STATEMENT CHECKS THAT THERE IS STILL SUFFICIENT
//* WORK SPACE ON THE VOLUME CONTAINING SYS1.PPOPTION TO RUN THE
//* INSTALLATION JOB STREAM.
//*
//WORK     DD  VOL=(PRIVATE,RETAIN,REF=SYS1.PPOPTION),SPACE=(CYL,4)
//SYSIN    DD  *
 LISTVTOC  VOL=2311=231100      * CHECK LIST DATA SET ALLOCATION *
/*
//OPTION EXEC  PGM=IEBUPDTE,PARM='NEW'
//*
//* ADD MACRO DEFINITION DEFINING RESIDENT LIBRARY OPTIONS REQUIRED TO
//* EXISTING SYS1.PPOPTION DATA SET.
//*
//SYSPRINT DD  SYSOUT=A
//SYSUT2   DD  DSN=SYS1.PPOPTION,VOL=(PRIVATE,RETAIN),DISP=OLD
//SYSIN    DD  *
./       ADD   NAME=PLRSLIB,LIST=ALL
      PLRSLIB  PLIBASE=(COMPLEX,ILC,EXTND),PLITASK=YES,PLRSHR=YES
      END
/*
//
```

Figure 8. (Part 2 of 2). Example of user specified JCL - resident library

Shared Library

The shared library feature enables certain preselected PL/I library modules to be shared between two or more PL/I programs. The job of creating the shared library is not part of the general installation procedure and can be carried out at any time after the installation of the resident library.

CREATING THE SHARED LIBRARY

To create the shared library you must run two punched card jobs. The first job, created by you, assembles a macro instruction in which you have specified the groups of modules required in the shared library, and the environment (tasking, non-tasking or both) in which you wish to use it. The second job, output from Job 1, linkage edits two or three modules to the link library, SYS1.LINKLIB, and two modules each to SYS1.PLIBASE and SYS1.PLITASK. A further job must then be performed at initial program load to permanently store the shared library in main storage (see "Shared Library in Main Storage") before using it. On completion of these jobs the shared library feature is available for use. For information on using the shared library see the publication: OS PL/I Optimizing Compiler: Programmer's Guide.

Job 1

The first step in the creation of the shared library is to specify the groups of modules that you require in the shared library (see the description of the PLRSHR macro instruction for groups of modules available). You then specify the groups of modules in the PLRSHR macro instruction as shown in figure 9. It is advisable to specify only frequently used modules to make best use of the shared library feature. Note that the housekeeping and basic error handling modules are always included in the shared library.

Note: When specifying the PLRSHR macro instruction you will either select JOBSTMT=NOTSUPPLIED or SUPPLIED. If you select NOTSUPPLIED a default job card will be generated as the first card in the job stream, as follows:

```
//PLRSHR JOB 'SHARED LIBRARY
  GENERATION' MSGLEVEL=1
```

However if you select SUPPLIED you add your own job card during Job 1, as shown in figure 9, before the job is run.

```
//SHLIB     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=2400,LABEL=(1,NL)
//SYSLIB    DD    DSN=PLI.SHRMAC,DISP=SHR
//SYSIN     DD    *
          PUNCH  'DETAIL HERE JOB CARD OF YOUR OWN CHOOSING'
          PLRSHR  MODES=(TASK,BASE),STORG=ERR,CONV=ARITH,
                  STRGS=BIT,IO=(LIST.INPUT,EDIT.OUTPUT),MFUNC=(SORT,DEBUG),
                  CONTL=(TASK.WAIT),JOBSTMT=SUPPLIED
  END
/*
```

Figure 9. Example job to specify the shared library

When you have selected and specified the groups of modules in the PLRSHR macro instruction punch out a job, similar to the one in figure 9. 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.

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 link library, SYS1.LINKLIB.

Note: Before the job is run, you must make sure that the cataloged procedure PLSRDEF, used to define the target libraries during the installation procedure of the resident library, is still on the system and defines the link library data set to be used.

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 card you supplied or a default job card. Start a reader to the SYSPUNCH data set and run Job 2.

The job will assemble and linkage edit the following modules:

- IBMBPSRA if (MODES = BASIC) is specified, and/or IBMTPSRA if (MODES = TASK) is specified, onto the basic and/or tasking subroutine libraries. 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.
- IBMBPSMA and IBMBPSLA if (MODES = BASE) is specified, and/or IBMTPSLA if (MODES = TASK) is specified, onto the link library data set specified in the PLRSDEF cataloged procedure. These shared library modules will consist of subroutines from the basic, or basic and tasking subroutine libraries, selected according to the options specified in the PLRSHR macro.

Shared Library in Main Storage

Before the shared library can be used it must be loaded permanently into main storage during initial program load. The shared library at this point consists of three reentrant load modules, and your installation must be able to make this type of module resident during initial program load. For more information on making reentrant load modules resident, refer to the publications: OS System Generation, and OS System Programmer's Guide.

Initial Program Load (IPL)

There are three possible combinations of load modules, these being:

- Basic shared library only - IBMBPSLA and IBMBPSMA.
- Tasking shared library only - IBMTPSLA and IBMBPSMA.

- Basic and tasking shared library - IBMBPSLA, IBMTPSLA, and IBMBPSMA.

To store any of these three combinations of modules permanently in main storage you must first build a new reentrant load module user list for the modules in the parameter library, SYS1.PARMLIB; and second, you must ensure that the use of this list by the system is specified during initial program load. You can give this user list a name of the form IEAIGGxx, where xx are any two digits other than 00. The following example of coding suggests how you create such a list by means of the IEBUPDTE utility program (see the publication: OS Utilities for more information); the method of specifying its use by the system is given in the following paragraph. For illustration purposes, the example has been given the user list name IEAIGG01 and a basic and tasking shared library is being used.

```
//BLDLIST  EXEC   PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD     SYSOUT=A
//SYSUT2   DD     DSN=SYS1.PARMLIB,
//                UNIT=2314,
//                VOL=(,RETAIN,SER=MVT111),
//                DISP=OLD
//SYSIN    DD     *
./         ADD    NAME=IEAIGG01,LIST=ALL
./         NUMBER NEW=01,INCR=02
  SYS1.LINKLIB (IBMBPSLA,IBMTPSLA,IBMBPSMA)
./    ENDUP
/*
```

The ADD statement identifies your shared library load modules, IBMBPSLA, IBMTPSLA, and IBMBPSMA in the new reentrant load module list (IEAIGG01). When the nucleus initialization program (NIP) requests the system parameters, the operator responds by specifying 'RAM=01' (the digits 01 correspond to the last two digits of the user list name IEAIGG01). This response causes your load module to be loaded into the relevant area of main storage.

If you require both the standard and shared library load module lists, you must instruct the operator to respond to the NIP request for system parameters by specifying 'RAM=00,01'. You should note the following points:

- If your installation standard reentrant load module list (IEAIGG00) refers to SYS1.LINKLIB and not SYS1.SVCLIB, you can add your module entry point names (IBMBPSLA, IBMTPSLA, and IBMBPSMA) to this list. In this case you do not need to create a new procedure library list; your shared library load module will be automatically stored in main storage at initial program load.
- If your installation standard reentrant load module list (IEAIGG00) does not refer to SYS1.LINKLIB, your system will need to have been generated with the ability to communicate with the operator during initial program load. This is because the operator must specify the use of the alternative list as described previously.

SHARED LIBRARY SYSTEM GENERATION MACRO

You use the PLRSHR system generation macro to create the shared library, choosing shared library operands as required.

The format of the PLRSHR shared library system generation macro is shown below. Items in upper case must be coded as shown, items in lower case should be coded as indicated by the operand specifications.

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

The following is a detailed description of the operands:

MODES = ([TASK][,BASE])

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 commoned into a single module for both tasking and non-tasking functions.

Note: MODES=BASE is the default.

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 in an AREA and assignment between AREAs will be made resident.

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

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 conversions.

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.

CONTL =
module associated with WAIT, PRIORITY and COMPLETION in a tasking and/or non-tasking 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 pseudo-variable in a tasking system.

PRTY - modules associated with the PRIORITY pseudo-variable 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=(TASK,BASE) is specified.

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

CPLN - modules associated with the COMPLETION pseudo-variable in a non-tasking system.

PRTY - modules associated with the PRIORITY pseudo-variable in a non-tasking system.

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 pseudo-variable 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 pseudo-variable SUBSTRING

STRING[.BIF][.PV]

the modules associated with the STRING built-in function and pseudo-variable will be made resident.

- BIF - the STRING built-in function module
- PV - the STRING pseudo-variable module

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 binary and fixed decimal elements.

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

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

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.

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 trigonometrical functions SIN, COS, TAN, SIND, COSD, TAND.

ATRIG - The trigonometrical functions ASIN, ACOS, ATAN, ATAND, ASIND, ACOSD

HYPER - the hyperbolic functions SINH and TANH

AHYPER - 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 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 explanation see RMATH SHORT)

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 binary or 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 function with COMPLEX SHORT precision floating point arguments will be made resident.

TRIG - the trigonometrical 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 trigonometrical 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 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 explanation of subgroup elements see CMATH SHORT)

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 modules associated with the FLOW option

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.

JOBSTMT =
to indicate whether the macro is to generate a JOB card or whether you will provide one by giving a PUNCH statement for, or placing a REPRO statement immediately preceding your JOB card 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 card will be generated.

//PLSHR JOB 'SHARED LIBRARY GENERATION'
MSGLEVEL=1
This option will be the default

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

READER'S COMMENT FORM

OS PL/I Optimizing Compiler:
System Information

SC33-0026-0

- How did you use this publication?

  As a reference source ..... ☐
  As a classroom text ..... ☐
  As a self-study text ..... ☐

- Based on your own experience, rate this publication . . .

	.....	.....	.....	.....	.....
As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation? .....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

- Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A.

YOUR COMMENTS PLEASE . . . .

This SRL manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note : Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Cut Along Line

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Department 813 (HP)

fold fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

OS PL/I Optimizing Compiler System Information Printed in U.S.A. SC33-0026-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]