High Level Assembler for z/OS & z/VM & z/VSE

Installation and Customization Guide

Version 1 Release 6

SC26-3494-05

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 189.

This edition applies to IBM High Level Assembler for z/OS & z/VM & z/VSE, Release 6, Program Number 5696-234 and to any subsequent releases until otherwise indicated in new editions. Make sure that you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality.

IBM welcomes your comments. For information on how to send comments, see "How to send your comments to IBM" on page xiv.

Contents

Figures

Tables

About this document

This document provides information needed to plan for, install, customize, maintain, and diagnose problems with High Level Assembler on z/OS®, z/VM®, z/VSE®, and Linux for System z® (zLinux).

Throughout this book, we use these indicators to identify platform-specific information:

- Prefix the text with platform-specific text (for example, "Under CMS...")
- Add parenthetical qualifications (for example, "(CMS)")
- A definition list, for example:

  z/OS Informs you of information specific to z/OS.

  z/VM Informs you of information specific to z/VM.

  z/VSE Informs you of information specific to z/VSE.

CMS is used in this manual to refer to Conversational Monitor System on z/VM.

Brief overview of High Level Assembler

High Level Assembler is an IBM® licensed program that helps you develop programs and subroutines using assembler language to provide functions not typically provided by other symbolic languages, such as COBOL, FORTRAN, PL/I, and C.

Who should use this document

This document is for system programmers and system administrators who plan for, install, customize, and maintain High Level Assembler on z/OS, z/VM, z/VSE, or zLinux.

It is also relevant to users who carry out diagnostic tasks on this product.

To use this document, you need to be familiar with the z/OS, z/VM, z/VSE, or zLinux operating systems, the publications that describe your system, and job control language (JCL) or EXEC processing.

Experienced users installing on z/OS

If you are installing on z/OS and are experienced in installing products with SMP/E, use the following fast path items:

1. DASD storage for target and distribution libraries is in Table 7 on page 4 and Table 8 on page 5.
2. DASD storage for SMP/E data sets is outlined in Table 5 on page 3 and Table 6 on page 4.
3. Carry out steps "Step 1: SMP/E considerations for installing High Level Assembler" on page 18 through to "Step 9: Run the installation verification program" on page 22.

Experienced users installing on z/VM

If you are installing on z/VM and are experienced in installing products with VMSES/E, use the following fast path items:

1. Refer to Table 25 on page 40 for DASD storage requirements.
2. Carry out steps "Step 1: Prepare to install High Level Assembler" on page 47 through to "Step 6: Put High Level Assembler into production" on page 59.

Experienced users installing on z/VSE

If you are installing on z/VSE and are experienced in installing products with MSHP, use the following fast path:

- Refer to Table 37 on page 81 and Table 38 on page 81 for DASD storage requirements.
- Carry out steps "Step 1: Back up the original system" on page 89 through to "Step 4: Verify the installation of High Level Assembler" on page 95.

Experienced users installing on zLinux

If you are installing on zLinux and are experienced in installing products on zLinux using RPM, carry out the procedures described in Chapter 17, "Installing High Level Assembler on zLinux," on page 109.

Terminology in this book

For simplicity, many of the full IBM product names are shortened to just the generic acronym. For example, MVS™ is used to mean z/OS.

List of APARs and PTFs in this book

This book includes the numbers for APARs and PTFs that are included in this product (see Appendix C, "High Level Assembler Service," on page 183). You might want to use this list, for example, to determine whether a fix that you have applied to the previous release is in this release that is to be installed.

To obtain current service recommendations and to identify current product service requirements, get the Preventive Service Planning (PSP) information or check with the IBM Support Center.

Other documentation you might need

The complete list of High Level Assembler publications and order numbers are listed on page 2, page 38 and page 80. Publications for related IBM products are in the "Bibliography" on page 191.

The High Level Assembler Web site, at

```
http://www.ibm.com/software/awdtools/hlasm
```

provides access to all High Level Assembler publications, in downloadable or directly viewable PDF format.

Syntax notation

Throughout this book, syntax descriptions use this structure:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ►►── symbol indicates the beginning of a statement.

  The ──► symbol indicates that the statement syntax is continued on the next line.

  The ►── symbol indicates that a statement is continued from the previous line.

  The ──►◄ indicates the end of a statement.

  Diagrams of syntactical units other than complete statements start with the ►── symbol and end with the ──► symbol.
- **Keywords** appear in uppercase letters (for example, ASPACE) or uppercase and lowercase (for example, PATHFile). They must be spelled exactly as shown. Lowercase letters are optional (for example, you could enter the PATHFile keyword as PATHF, PATHFI, PATHFIL, or PATHFILE).

  Variables appear in all lowercase letters in a special typeface (for example, *integer*). They represent user-supplied names or values.

- If punctuation marks, parentheses, or such symbols are shown, they must be entered as part of the syntax.
- Required items appear on the horizontal line (the main path).

```
►►──INSTRUCTION──required item─────────────────────────────────────────────►◄
```

- Optional items appear below the main path. If the item is optional and is the default, the item appears above the main path.

```
                ┌─default item──┐
►►──INSTRUCTION─┼───────────────┼──────────────────────────────────────────►◄
                └─optional item─┘
```

- When you can choose from two or more items, they appear vertically in a stack.

  If you **must** choose one of the items, one item of the stack appears on the main path.

```
►►──INSTRUCTION──┬─required choice1─┬──────────────────────────────────────►◄
                 └─required choice2─┘
```

  If choosing one of the items is optional, the whole stack appears below the main path.

```
►►──INSTRUCTION──┬──────────────────┬──────────────────────────────────────►◄
                 ├─optional choice1─┤
                 └─optional choice2─┘
```

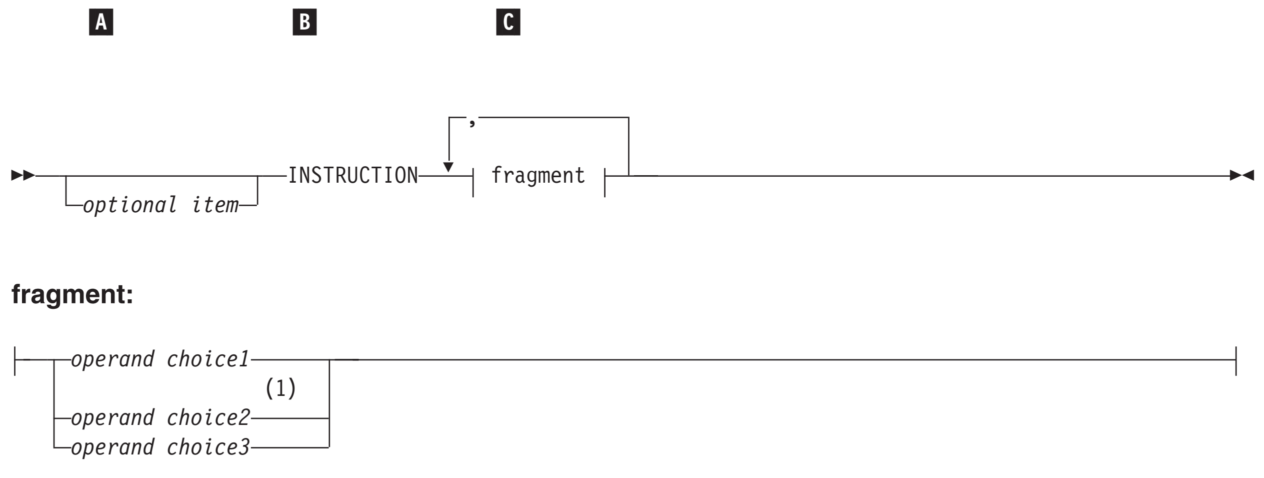
- An arrow returning to the left above the main line indicates an item that can be repeated. When the repeat arrow contains a separator character, such as a comma, you must separate items with the separator character.

```
                 ┌─,───────────────┐
                 ▼                 │
►►──INSTRUCTION────repeatable item─┴───────────────────────────────────────►◄
```

  A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

Format

The following example shows how the syntax is used.



Notes:

1 *operand choice2* and *operand choice3* must not be specified together

A The item is optional, and can be coded or not.

B The INSTRUCTION key word must be specified and coded as shown.

C The item referred to by "fragment" is a required operand. Allowable choices for this operand are given in the fragment of the syntax diagram shown below "fragment" at the bottom of the diagram. The operand can also be repeated. That is, more than one choice can be specified, with each choice separated by a comma.

How to send your comments to IBM

If you especially like or dislike anything about this book, feel free to send us your comments.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information that is in this book and to the way in which the information is presented. Speak to your IBM representative if you have suggestions about the product itself.

When you send us comments, you grant to IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

You can get your comments to us quickly by sending an e-mail to **idrcf@hursley.ibm.com**. Alternatively, you can mail your comments to:

User Technologies,
IBM United Kingdom Laboratories,
Mail Point 095, Hursley Park,
Winchester, Hampshire,
SO21 2JN, United Kingdom

Please ensure that you include the book title, order number, and edition date.

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative

- Call IBM technical support
- Visit the IBM support web page

Summary of changes

Date of Publication
August 2013

Form of Publication
Sixth Edition, SC26-3494-05

There are no major changes to the installation and customization procedures for High Level Assembler with release 6.

A new section is included, describing the installation and usage of High Level Assembler on zLinux.

Installation instructions for High Level Assembler release 6 on VM are provided for z/VM only. No reference is made to VM/ESA®, as VM/ESA is no longer a supported product.

Any reference to obsolete DASD devices has been removed from the z/VSE chapters.

Changes in this edition of the book are indicated by a vertical bar in the left margin.

For information about changes to the High Level Assembler product in release 6, refer to *HLASM General Information*.

Chapter 1. Planning for installing High Level Assembler on z/OS

This section contains the following planning information to help you properly install High Level Assembler on z/OS:
- Worksheet
- What You receive with High Level Assembler
- What You need to install High Level Assembler
- FMIDs deleted
- Installing with CBPDO
- Planning to use SMP/E
- Planning to change installation Jobs
- Program support
- Program and service level information
- Publications useful during installation

What you receive with High Level Assembler

You receive the following when you order High Level Assembler for z/OS:

FMID	Feature Number	System Name
HMQ4160	5892	z/OS

Distribution media

High Level Assembler is distributed on the following:
- 3480 tape cartridge

The cartridge contains all the programs and data needed for installation. The cartridge is in SMP/E relative file format. The first file contains the SMP/E modification control statements. Subsequent files contain IEBCOPY, unloaded partitioned data sets, which SMP/E processes.

Basic material

Table 1 describes the cartridge. Table 2 describes the file content of the cartridge.

Table 1. Basic material: program cartridge

Medium	Feature Number	Physical Volume	External Label Identification	VOLSER
3480 cart.	5892	1	HLASM V1R6 for MVS	MQ4160

Table 2. Program cartridge: file content

Volser	File	Name	ORG	RECFM	LRECL	BLK SIZE
MQ4160	1	SMPMCS	SEQ	FB	80	6400
	2	IBM.HMQ4160.F1	PDS	FB	80	8800
	3	IBM.HMQ4160.F2	PDS	FB	80	8800
	4	IBM.HMQ4160.F3	PDS	U	0	6144
	5	IBM.HMQ4160.F4	PDS	FB	80	8800

The RECFM, LRECL, and BLK SIZE reflect the original values of the partitioned data sets (relative files) before being unloaded by the IEBCOPY utility to tape.

Note: If you are installing High Level Assembler using the MVS Custom-Built Product Delivery Offering (CBPDO) (5751-CS3), some of the information in these figures may not be valid. Consult the CBPDO documentation for actual values.

Optional material

There are no optional machine-readable materials for High Level Assembler.

Program publications and softcopy

This section identifies the publications for High Level Assembler.
- *HLASM Licensed Program Specifications* GC26-4944
- *HLASM Installation and Customization Guide* SC26-3494
- *HLASM Language Reference* SC26-4940
- *HLASM Programmer's Guide* SC26-4941
- *HLASM General Information* GC26-4943

For a list of books for related products, see "Bibliography" on page 191.

What you need to install High Level Assembler

This section identifies the system requirements for installing High Level Assembler. You need to plan for two different system environments:
- *Target system*—the system comprising the set of libraries that are updated by SMP/E and system utilities during installation
- *Driving system*—the system on which the jobs are run to install the program

The machine-readable components for High Level Assembler are installed in target and distribution libraries.
- *Target libraries* are the data sets in which the run-time copy of High Level Assembler is stored. These data sets include the executable program code and any other components used during execution, such as sample JCL and messages.
- *Distribution libraries* are the data sets in which additional copies of these components are stored. The copy of High Level Assembler kept in the distribution libraries can be kept at a different service level than that in the target libraries. SMP/E can reconstruct the target libraries in whole or in part from the distribution libraries. Therefore, service or user modifications can be removed from the run-time copy of High Level Assembler

In many cases, the same system can be used as both a driving system and a target system. However, you may want to set up a clone of your system to use as a target system by making a separate IPL-able copy of the running system. The clone should include copies of all system libraries that SMP/E updates, copies of the SMP/E CSI data sets that describe the system libraries, and your PARMLIB and PROCLIB.

Some cases where two systems should be used include the following:
- When installing a new level of a product that is already installed, the new product will delete the old one. By installing onto a separate target system, you can test the new product while still keeping the old one in production.
- When installing a product that shares libraries or load modules with other products, the installation can disrupt the other products. Installing onto a test system or clone will allow you to assess these impacts without disrupting your production system.

Required and optional software for the target system

This section describes the other products that must be installed on the target system in order to install and use High Level Assembler.

High Level Assembler runs on z/OS with the required licensed programs listed in Table 3.

You should install the **minimum release listed or any subsequent release** for all required licensed programs your site needs.

Table 3. Required programs

Required Licensed Program	Minimum Version Supported
z/OS	Version 1 Release 7
SMP/E	SMP/E for z/OS V3.4.0

Check with the IBM Support Center for any PTFs you might need to apply.

DASD storage required for the target system

High Level Assembler libraries can reside on any currently supported DASD.

Table 4 lists the total space required for each type of library. The values are for a 3390 DASD.

Table 4. Total DASD space required by High Level Assembler

Library Type	Total Space Required
Target	72 Tracks
Distribution	76 Tracks

Table 5 gives an estimate of the storage requirements for SMP/E work data sets, for High Level Assembler.

Table 5. Storage requirements for SMP/E work data sets

Library DDNAME	TYPE	ORG	RECFM	LRECL	No. of 3390 Trks	No. of DIR Blks
SMPWRK1	S	PDS	FB	80	30	80
SMPWRK2	S	PDS	FB	80	30	80
SMPWRK3	S	PDS	FB	80	180	80
SMPWRK4	S	PDS	FB	80	30	80
SMPWRK6	S	PDS	FB	80	30	80
SYSUT1	U	SEQ	--	--	30	0
SYSUT2	U	SEQ	--	--	30	0
SYSUT3	U	SEQ	--	--	30	0
SYSUT4	U	SEQ	--	--	30	0

Notes:

1. The data set sizes specified contain 15% extra space. You may wish to revise these numbers based on your plans for adding additional function or service.
2. IBM recommends use of system determined blocksizes for efficient DASD utilization for all non-RECFM U data sets. For RECFM U data sets, IBM recommends a blocksize of 32760, which is the most efficient from a performance and DASD utilization perspective.

   If you choose not to use system determined blocksizes, use the blocksizes and numbers of blocks specified to allocate the data sets. Data sets can be reblocked to a larger size. The maximum allowable blocksize depends on the type of DASD on which the dataset will reside.
3. Abbreviations used for the data set type are:

   U Unique data set used by only the FMIDs listed. In order to determine the correct storage needed for this data set, this table provides all required information; no other tables (or program directories) need to be referenced for the data set size.

   S Shared data set used by more than the FMIDs listed. In order to determine the correct storage needed for this data set, the storage size given in this table needs to be added to other tables (perhaps in other program directories). If the data set already exists, it must have enough free space to accommodate the storage size given in this table.

   If you currently have a previous release of this product installed in these libraries, the installation of this release will delete the old one and reclaim the space used by the old release and any service that had been installed. You can determine whether these libraries have enough space by deleting the old release with a dummy function, compressing the libraries, and comparing the space requirements with the free space in the libraries.

   For more information about the names and sizes of the required data sets, refer to Table 7 and Table 8 on page 5.

Table 6 provides an estimate of the storage needed in the SMP/E data sets for High Level Assembler. The estimates must be added to those of any other programs and service being installed to determine the total additional storage requirements.

Table 6. Storage requirements for SMP/E data sets

Library DDNAME	T Y P E	O R G	R E C F M	L R E C L	No. of 3390 Trks	No. of DIR Blks
SMPMTS	S	PDS	FB	80	30	5
SMPPTS	S	PDS	FB	80	30	5
SMPSCDS	S	PDS	FB	80	30	5
SMPSTS	S	PDS	FB	80	30	5

Table 7 and Table 8 on page 5 list the target and distribution libraries and their attributes required to install High Level Assembler. The storage requirements of High Level Assembler must be added to the storage required by other programs having data in the same library.

Table 7. Storage requirements for High Level Assembler target libraries

Library DDNAME	Member Type	Target Volume	T Y P E	O R G	R E C F M	L R E C L	No. of 3390 Trks	No. of DIR Blks
SASMMOD1	LMOD	ANY	U	PDS	U	0	16	3
SASMSAM1	SAMPLES PROCS	ANY	U	PDS	FB	80	36	2
SASMMAC1	MACRO	ANY	U	PDS	FB	80	20	1

Table 8. Storage requirements for High Level Assembler distribution libraries

Library DDNAME	T Y P E	O R G	R E C F M	L R E C L	No. of 3390 Trks	No. of DIR Blks
AASMMOD1	U	PDS	U	0	24	14
AASMSAM1	U	PDS	FB	80	32	2
AASMMAC1	U	PDS	FB	80	20	1

You can access the supplied sample JCL after the SMP/E RECEIVE step in the install process. Alternatively, you can copy the sample JCL directly from the installation tape into a temporary library. If you use this method, use the data set requirements shown in Table 9.

Table 9. Storage requirements for temporary library

Data Set Name	T Y P E	O R G	R E C F M	L R E C L	No. of 3390 Trks	No. of DIR Blks
JCL	U	SEQ	FB	80	30	4

FMIDs deleted

Installing High Level Assembler may result in the deletion of other FMIDs. To see what FMIDs will be deleted, examine the ++VER statement in the product's SMPMCS.

If you do not wish to delete these FMIDs at this time, you must install High Level Assembler into separate SMP/E target and distribution zones.

Note: These FMIDs will not automatically be deleted from the Global Zone. Consult the SMP/E manuals for instructions on how to do this.

Installing with CBPDO

If you are installing High Level Assembler with an MVS Custom-Built Product Delivery Offering (CBPDO) (5751-CS3), use the RCVPDO job in the CBPDO RIMLIB data set provided with the CBPDO to receive the product and service for this product. Any additional installation instructions should be obtained from the High Level Assembler documentation. However, before installing High Level Assembler, check with your IBM Support Center or use either Information/Access or SoftwareXcel Extended to see whether there is additional service information you need.

Planning to use SMP/E

You must decide how to use SMP/E to install High Level Assembler. Some points to consider are:

1. Choose to use new global, target, and distribution zones.
2. Choose to use existing global and new target and distribution zones.
3. Choose to use existing global, target, and distribution zones.
4. Use the supplied sample jobs or use SMP/E dialogs to RECEIVE, APPLY and ACCEPT. See Chapter 3, “Installing High Level Assembler on z/OS,” on page 17.

Planning to change installation jobs

Each sample job has instructions on how it is to be modified, to meet the requirements and standards at your site. Refer to these standards as needed for determining the values for these parameters.

Publications useful during installation

The publications listed in Table 10 may be useful during the installation of High Level Assembler for z/OS.

Table 10. z/OS publications

Publication Title	Form Number
SMP/E Messages, Codes, and Diagnosis	GA22-7770
SMP/E Commands	SA22-7771
SMP/E Reference	SA22-7772
SMP/E User's Guide	SA22-7773

Program support

This section describes the IBM support available for High Level Assembler.

Program services

Contact your IBM representative for specific information about available program services.

Preventive Service Planning

Before installing High Level Assembler, you should review the current Preventive Service Planning (PSP) information. If you obtained High Level Assembler as part of a CBPDO, there is HOLDDATA and PSP information included on the CBPDO tape.

If you obtained High Level Assembler on a product tape, or if the CBPDO is more than two weeks old when you install it, you should contact the IBM Support Center or use S/390® SoftwareXcel to obtain the current “PSP Bucket”.

PSP Buckets are identified by UPGRADEs, which specify product levels, and SUBSETs, which specify the FMIDs for a product level. The UPGRADE and SUBSET values for High Level Assembler for z/OS are:

Table 11. PSP upgrade and subset ID

UPGRADE	SUBSET	Description
HLASM160	HMQ4160	HLASM MVS

Statement of support procedures

Report any difficulties you have using this program to your IBM Support Center. If an APAR is required, the Support Center will provide the address to which any needed documentation can be sent.

Table 12 identifies the component IDs (COMPID) for High Level Assembler for z/OS.

Table 12. Component IDs

FMID	COMPID	Component Name	RETAIN® Release
HMQ4160	569623400	MVS HIGH LEVEL ASM	160

Program and service level information

This section identifies the program and any relevant service levels of High Level Assembler. The program level refers to the APAR fixes incorporated into the program. The service level refers to the PTFs integrated. Information about the cumulative service tape is also provided.

Program level information

A list of APAR fixes against previous releases of High Level Assembler that have been incorporated into this release is shown in Appendix C, “High Level Assembler Service,” on page 183.

Service level information

No PTFs against this release of High Level Assembler have been incorporated into the product tape.

Cumulative service tape

A cumulative service tape, containing PTFs not incorporated into this release, might be included with this program. Installation instructions for cumulative service tapes can be found in the SMP/E publications.

If you received this product as part of a CBPDO or a ProductPac®, PTFs not incorporated into this release are provided on the tape, and a separate cumulative service tape will not be provided.

Chapter 2. Planning for customizing High Level Assembler on z/OS

This chapter provides information for planning the customization of High Level Assembler on z/OS. It includes:
- Deciding whether and what to customize
- Impact of making SMP/E use High Level Assembler as its assembler
- Planning to customize the IBM-supplied default option values
- Planning to customize the IBM-supplied default user exits
- Planning to customize the cataloged procedures
- Planning to install High Level Assembler into a link pack area

Deciding whether and what to customize

You need to consider whether the IBM-supplied values that come with High Level Assembler suit the needs of your site. These values control such features as:
- SMP/E assembler use
- Selecting multicultural support
- Assembler options
- Default file names
- User exits
- Sample procedures

Make sure that High Level Assembler serves the needs of the application programmers at your site. Confer with them while you evaluate the customization options for High Level Assembler, particularly those concerning High Level Assembler options that are also available to the application programmers. This ensures that the modifications you make best support the application programs being developed at your site.

The information in this chapter helps you plan your customization. See Chapter 4, “Customizing High Level Assembler on z/OS,” on page 25 for the actual customization procedure.

SMP/E and High Level Assembler

The default SMP/E OPTIONS entry for the assembler utility is ASMA90. SMP/E runs authorized, therefore all utility programs called by SMP/E must reside in an authorized library. Consequently, unless this default has been changed at your site to an alternative product, you must ensure High Level Assembler resides in an Authorized Program Facility (APF) authorized library.

Furthermore, if access to High Level Assembler is to be restricted, the z/OS Security Server must be used to control its execution.

Selecting multicultural support

When customizing High Level Assembler, you can choose which language you want to use for diagnostic messages. Languages available are English (mixed case or upper case), German, Spanish and Japanese. You specify the language by means of the LANGUAGE option. See page “LANGUAGE” on page 154 for information about the LANGUAGE option.

The following combinations are possible:

English Uppercase
Diagnostic messages and listing headings printed in uppercase English.

English Mixed Case
Diagnostic messages and listing headings printed in mixed case English.

German
Diagnostic messages in German and listing headings printed in mixed case English.

Japanese
Diagnostic messages in Kanji and listing headings printed in uppercase English.

Spanish
Diagnostic messages in Spanish and listing headings printed in mixed case English.

Planning to customize High Level Assembler options

High Level Assembler can be customized with a large range of options and DDNAMES for users. When customized they become the default options and DDNAMES.

Why do it

The High Level Assembler options and DDNAMES should be reviewed to assess the required defaults for your site. Worksheets are provided for planning purposes.

Choices to make now

The following worksheets help you plan and code the options and DDNAMES appropriate for your site. To complete the worksheets, fill in the **Enter Selection** column.

Table 13. Worksheet options

Object	Enter Selection	IBM-supplied default	Description
ADATA	__________	NO	page 143
ADEXIT	__________	no exit specified	page 144
ALIGN	__________	YES	page 144
ALIGNWARN	__________	YES	page 145
ASA	__________	nothing specified	page 145
BATCH	__________	YES	page 145
CODEPAGE	__________	047C	page 146
COMPAT	__________	NO	page 146
CONTWARN	__________	YES	page 147
DBCS	__________	NO	page 148
DECK	__________	NO	page 148
DELETE	__________	no options deleted	page 149
DSECT	__________	NO	page 150
DXREF	__________	YES	page 150
ESD	__________	YES	page 151
EXLITW	__________	YES	page 151
FLAG	__________	0	page 151
FOLD	__________	NO	page 152
GOFF	__________	NO	page 152
GOFFADATA	__________	NO	page 152
IMPLENWARN	__________	NO	page 153
INEXIT	__________	no exit specified	page 153

Table 13. Worksheet options (continued)

Object	Enter Selection	IBM-supplied default	Description
INFO	__________	NO	page 154
LANGUAGE	__________	EN	page 154
LIBEXIT	__________	no exit specified	page 155
LIBMAC	__________	NO	page 155
LIMIT	__________	NO	page 156
LINECOUNT	__________	60	page 156
LIST	__________	121	page 157
MACHINE	__________	no default, see OPTABLE	page 158
MACHINELIST	__________	NO	page 159
MAP	__________	YES	page 159
MXREF	__________	SOURCE	page 160
OBJECT	__________	YES	page 160
OBJEXIT	__________	no exit specified	page 161
OPTABLE	__________	UNI	page 161
OPTABLELIST	__________	NO	page 163
PAGE0WARN	__________	NO	page 163
PCONTROL	__________	NO	page 164
PESTOP	__________	NO	page 165
PROFILE	__________	NO	page 166
PROFMEM	__________	ASMAPROF	page 166
PRTEXIT	__________	no exit specified	page 167
PUSHWARN	__________	YES	page 167
RA2	__________	NO	page 167
RECORDINFO	__________	YES	page 168
RENT	__________	NO	page 168
RLD	__________	YES	page 169
RXREF	__________	YES	page 169
SECTALGN	__________	8	page 169
SIZE	__________	MAX	page 170
STORAGE	__________	nothing specified	page 171
SUBSTRWARN	__________	NO	page 171
SUPRWARN	__________	NO	page 172
SYSPARMV	__________	none specified	page 172
TERM	__________	NO	page 173
TEST	__________	NO	page 173
THREAD	__________	YES	page 173
TRANSLATE	__________	NO	page 174
TRMEXIT	__________	no exit specified	page 174
TYPECHECK	__________	(MAGNITUDE, REGISTER)	page 175

Table 13. Worksheet options (continued)

Object	Enter Selection	IBM-supplied default	Description
USING0WARN	__________	YES	page 175
WARN	__________	15	page 176
WORKFILE	__________	NO	page 176
XOBJADATA	__________	NO	page 177
XOBJECT	__________	NO	page 177
XREF	__________	(SHORT,UNREFS)	page 178

Table 14. Worksheet: DDNAMEs

DDNAME	EnterSelection	IBM-SuppliedDefault	Description
ADATA	__________	SYSADATA	page 179
IN	__________	SYSIN	page 179
LIB	__________	SYSLIB	page 180
LIN	__________	SYSLIN	page 180
OPT	__________	ASMAOPT	page 180
PRINT	__________	SYSPRINT	page 181
PUNCH	__________	SYSPUNCH	page 181
TERM	__________	SYSTERM	page 181
UT1	__________	SYSUT1	page 182

Planning to customize the user exits

Several exits are provided with the High Level Assembler, they are described in the *HLASM Programmer's Guide*. The exit types are:
- Source exit
- Listing exit
- ADATA exit

In order to help you customize your exits, sample source is provided, as follows:

ASMAXINV
Sample exit to read variable length records

ASMAXPRT
Sample listing exit

To support the ADATA exit the following are provided:

ASMAXFSK
Sample skeleton SYSADATA filter routine

ASMAXFMT
Sample filter management table

ASMAXFLU
Sample filter module to dump ADATA records

ASMAXADT
Sample ADATA exit to load filter routine

ASMAXADR
Sample exit to reformat ADATA records from new to old format

ASMAXADC
Sample exit to control ADATA record output

These samples are distributed in the ASM.SASMSAM1 target library.

Note: If User Exits from a previous release are being used, these exits may need to be reassembled for this release.

Planning to customize cataloged procedures

High Level Assembler provides four cataloged procedures to:
- Assemble a program
- Assemble and link-edit a program
- Assemble, link-edit, and run a program
- Assemble a program and use the loader to run it

Why do it

Providing procedures that can assemble, link-edit, or link-and-go allow the user to provide their assembler source, and select the appropriate procedure and run their job. It minimizes the amount of user JCL required and provides a standard convention to process assembler source.

These procedures provide the user with standard JCL that is tailored to your installation requirements. The user does not need to keep track of JCL changes.

The cataloged procedures distributed by IBM execute in as many z/OS environments as possible. You should review these procedures and optimize them for your environment.

In general, installation conventions stipulate the options that you include in the PARM, UNIT, and SPACE parameters of the cataloged procedures.

Choices to make now

Four standard procedures are supplied; refer to Table 15 for their descriptions.

Table 15. Supplied procedures

Procedure	Description
ASMAC	Assembles only
ASMACL	Assembles and link-edits program
ASMACLG	Assembles, link-edits program and runs program
ASMACG	Assembles and uses the loader to run program

If High Level Assembler Release 1 has previously been used, users might have become accustomed to using the old procedures names of HLASMC, HLASMCG, HLASMCLG, and HLASMCL. In order to make the transition from this previous release, in the ASM.SASMSAM1 library, these old procedure names appear as aliases for the sample supplied procedures, as shown in Table 16.

Table 16. New and old procedure names

New Name	Old Name
ASMAC	HLASMC
ASMACL	HLASMCL
ASMACLG	HLASMCLG

Table 16. New and old procedure names (continued)

New Name	Old Name
ASMACG	HLASMCG

Example

Figure 1 shows JCL that invokes a sample procedure to assemble a small assembler program.

```
//ASMAPTST JOB  <JOB CARD PARAMETERS>
//ASMAPTST EXEC PROC=ASMAC
//SYSPRINT DD   SYSOUT=*
//STEPLIB  DD  DSN=#hlq.SASMMOD1,
//             DISP=SHR
//SYSIN    DD  *
TEST       CSECT
           XR   15,15
           BR   14
           END
/*
```

Figure 1. Use of sample procedure

This example is supplied as ASMAPTST in ASM.SASMSAM1.

Planning to install in the link pack area

This section describes the reasons why you should place High Level Assembler modules in common storage, and lists the modules that can you can place there.

Why do it

You might want to make some load modules resident in a link pack area in order to minimize the search path length when High Level Assembler is run.

All assembler load modules are reentrant and eligible for placement in the link pack area (LPA) and the modifiable link pack area (MLPA). The LPA and the MLPA are areas of storage that are shared by all users. By placing modules in these areas, you reduce the amount of storage needed in the user region.

By sharing the information you can:
- Save on real storage.
- Reduce I/O activity to the LOADLIB and page data sets.
- Reduce elapsed time of jobs.

Choices to make now

Table 17 is a list of module names, their link-edit attributes and approximate sizes.

Table 17. Module information

Module name	Amode	Rmode	Other	Linkedit	Attributes	Approx. Size
ASMA90	31	ANY	RENT	REUS	REFR	428K
ASMADOPT	31	ANY	RENT	REUS	REFR	2K
ASMAMUE	31	ANY	RENT	REUS	REFR	22K
ASMAMDE	31	ANY	RENT	REUS	REFR	26K

Table 17. Module information (continued)

Module name	Amode	Rmode	Other	Linkedit	Attributes	Approx. Size
ASMAMES	31	ANY	RENT	REUS	REFR	25K
ASMAMJP	31	ANY	RENT	REUS	REFR	25K
ASMALTAS	31	ANY	RENT	REUS	REFR	1K
ASMAINFO	31	ANY	RENT	REUS	REFR	42K
ASMA0474	31	ANY	RENT	REUS	REFR	2K
ASMA0475	31	ANY	RENT	REUS	REFR	2K
ASMA0476	31	ANY	RENT	REUS	REFR	2K
ASMA0477	31	ANY	RENT	REUS	REFR	2K
ASMA0478	31	ANY	RENT	REUS	REFR	2K
ASMA0479	31	ANY	RENT	REUS	REFR	2K
ASMA047A	31	ANY	RENT	REUS	REFR	2K
ASMA047B	31	ANY	RENT	REUS	REFR	2K
ASMA047C	31	ANY	RENT	REUS	REFR	2K
ASMADOP	31	ANY	RENT	REUS	REFR	59K

If you prefer to move some or all of the load modules, move them with a USERMOD that has a `++MOVE` statement.

Placing assembler modules in the LPA and MLPA

For information about placing modules into the LPA and MLPA, see *z/OS MVS Initialization and Tuning Reference*.

Chapter 3. Installing High Level Assembler on z/OS

This chapter describes the installation method and the step-by-step procedures to install and to activate the functions of High Level Assembler.

Notes:

- If you want to install High Level Assembler into its own SMP/E environment, consult the SMP/E manuals for instructions on creating and initializing the SMPCSI and the SMP/E control data sets.
- Sample jobs have been provided to help perform some or all of the installation tasks. The SMP/E jobs assume that all DDDEF entries required for SMP/E execution have been defined in the appropriate zones.
- The SMP/E dialogs may be used instead of the sample jobs to accomplish the SMP/E installation steps.

Overview of installation

You install this release of High Level Assembler using the SMP/E RECEIVE, APPLY, and ACCEPT commands. You can do this by using SMP/E dialogs or running batch jobs.

Modifying High Level Assembler jobs

The installation jobs that IBM provides do not necessarily follow the conventions (such as file-naming conventions) for your site. You might need to modify them. Descriptions of possible modifications appear in the comments in the sample jobs.

If using a CBPDO

If you obtained High Level Assembler as part of a CBPDO, you can use the RCVPDO job found in the CBPDO RIMLIB data set to RECEIVE the High Level Assembler FMID as well as any service, HOLDDATA, or preventive service planning (PSP) information included on the CBPDO tape. For more information, refer to the documentation included with the CBPDO.

Installation checklist

Table 18 lists the steps and associated jobs to install High Level Assembler. The remaining sections in this chapter describe each step. You can use Table 18 as a checklist.

Table 18. Summary of steps for installing High Level Assembler

Step	Description	Installation Job	Page
_1	SMP/E Considerations for Installing High Level Assembler		18
_2	Copy the Sample JCL from the Product Tape	ASMWCOPY	19
_3	Set up the ISPF Editor Macro (optional)	ASMWEDIT	19
_4	Allocate and Initialize the SMP/E Data Sets (optional)	ASMWSMPE	19
_5	Perform SMP/E Receive	ASMWRECV	20
_6	Allocate Target and Distribution Libraries	ASMWALOC	21
_7	Create DDDEF Entries	ASMWDDEF	21
_8	Perform SMP/E Apply	ASMWAPLY	21
_9	Run the Installation Verification Program	ASMWIVP	22
_10	Perform SMP/E Accept	ASMWACPT	22

If you have previously installed an earlier release of High Level Assembler, note that the names of the sample jobs have changed from previous releases.

Step 1: SMP/E considerations for installing High Level Assembler

This release of High Level Assembler is installed using the SMP/E RECEIVE, APPLY, and ACCEPT commands. The SMP/E dialogs may be used to accomplish the SMP/E installation steps.

SMP/E environment

All SMP/E installation jobs provided assume that all necessary DD statements for the execution of SMP/E are defined using DDDEFs.

Sample jobs are provided on the distribution tape to help you install High Level Assembler. After the RECEIVE step has been completed, the sample jobs can be found in SMPTLIB: **IBM.HMQ4160.F1**. Make a copy of these jobs in your own library and modify them to use during the installation of High Level Assembler. The sample jobs are:

ASMWSMPE
Sample job to define an SMP/E environment

ASMWRECV
Sample RECEIVE job

ASMWALOC
Sample job to allocate target and distribution libraries

ASMWDDEF
Sample job to define High Level Assembler DDDEFs

ASMWAPLY
Sample APPLY job

ASMWIVP
Sample install verification job

ASMWACPT
Sample ACCEPT job

ASMWEDIT
Sample ISPF edit macro

In the sample SMP/E jobs provided, the name of the SMP/E CSI is `#globalcsi`. The global zone name in the SMP/E CSI is `GLOBAL`. The distribution zone name is `#dzone`. The target zone name is `#tzone`. The sample jobs should be updated to reflect the CSI and zone names used at your site. Refer to the instructions in the sample jobs for other changes you may need to make.

SMP/E options subentry values

The recommended values for some SMP/E CSI subentries are shown in Table 19. Use of values lower than these may result in failures in the installation process. DSSPACE is a subentry in the GLOBAL options entry. PEMAX is a subentry of the GENERAL entry in the GLOBAL options entry. Refer to the SMP/E manuals for instructions on updating the global zone.

Table 19. SMP/E options subentry values

SUB-ENTRY	Value	Comment
DSSPACE	(300,150,250)	Space Allocation for SMPTLIB data sets
PEMAX	SMP/E Default	IBM recommends using the SMP/E default for PEMAX.

Step 2: Copy the sample JCL from the product tape

This step can be bypassed if you plan to use the sample jobs from the SMPTLIB: **IBM.HMQ4160.F1** after the Receive Step.

The following sample JCL will copy the High Level Assembler jobs from the tape. Add a job card and modify the parameters in boldface to uppercase values to meet your site's requirements before submitting.

```
//COPY   EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=A
//IN     DD DSN=IBM.HMQ4160.F1,UNIT=#tape,VOL=SER=MQ4160,
//       DISP=(OLD,KEEP),LABEL=(2,SL)
//OUT    DD DSN=#hlq.JCL,
//       DISP=(NEW,CATLG,DELETE),
//       RECFM=FB,LRECL=80,BLKSIZE=0,
//       SPACE=(8800,(120,40,4)),
//       UNIT=#unit,
//       VOL=SER=#dvol
//SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SYSIN DD *
  COPY INDD=IN,OUTDD=OUT
/*
```

Figure 2. Load installation jobs

In this sample, **#tape** is the unit value matching the product cartridge, **#hlq.JCL** is the name of the data set where the sample jobs will reside, **#dvol** is the volume serial of the DASD device where the data set will reside, and **#unit** is the DASD unit type of the volume.

This sample job is also supplied as sample ASMWCOPY in SMPTLIB: IBM.HMQ4160.F1.

Step 3: Set up ISPF editor macro (optional)

To aid you in making changes to the SMP/E installation jobs (ASMWACPT, ASMWALOC, ASMWAPLY, ASMWDDEF, ASMWIVP, ASMWRECV, and ASMWSMPE), an ISPF editor macro, ASMWEDIT, is supplied and is copied to your output data set.

This macro lets you substitute correct values for all of the required variables in those jobs, instead of making the changes repeatedly in each job.

Edit the macro ASMWEDIT and provide the correct values. Then copy it to any data set in your TSO logon procedure SYSEXEC concatenation. Consult the instructions in the macro for more information.

Step 4: Allocate and initialize the SMP/E data sets (optional)

You can install HLASM R6 in the same SMP/E zone as z/OS V1.6.0 (or later), or in a different zone.

If you install into existing SMP/E data sets, make sure that you have enough space.

If you plan to install into an existing zone, the cluster should have already been allocated and primed. You can go on to the next step to perform an SMP/E RECEIVE.

To install into a new zone, use the ASMWSMPE sample job to allocate and prime the SMPCSI. Consult the instructions in the sample job for more information.

Expected Return Codes and Messages: This job should issue a return code of zero and no error messages.

Step 5: Perform SMP/E receive

There are two methods of receiving High Level Assembler:

- The interactive capability of SMP/E described in "Method 1: Receive using SMP/E dialogs."
- The batch procedure described in "Method 2: Receive using a batch job."

Method 1: Receive using SMP/E dialogs

Select the Command Generation option from the SMP/E Primary Option menu.

Supply the following information as you progress through subsequent command generation panels for the RECEIVE command:

Name of your CSI
____

Zone name
Global

HOLDDATA
No

All No

Select Yes

FMID you are installing
HMQ4160

Data set name
____

Volume serial
____

Unit ____________________

Label Type
____

File ____________________

Expected Return Codes and Messages: This should issue a return code of zero and no error messages.

Method 2: Receive using a batch job

Edit and submit job ASMWRECV to receive High Level Assembler. Consult the instructions in the sample job for more information.

Note: If you obtained High Level Assembler as part of a CBPDO, you can use the RCVPDO job found in the CBPDO RIMLIB data set to RECEIVE the High Level Assembler FMID as well as any service, HOLDDATA, or preventive service planning (PSP) information included on the CBPDO tape. For more information, refer to the documentation included with the CBPDO.

Expected Return Codes and Messages: This job should issue a return code of zero and no error messages.

Step 6: Allocate target and distribution libraries

Edit and submit job ASMWALOC to allocate the SMP/E target and distribution libraries for High Level Assembler. Consult the instructions in the sample job for more information.

Expected Return Codes and Messages: This job should issue a return code of zero and no error messages.

Step 7: Create DDDEF entries

Edit and submit job ASMWDDEF to create DDDEF entries for the SMP/E target and distribution libraries for High Level Assembler. Consult the instructions in the sample job for more information.

Expected Return Codes and Messages: This job should issue a return code of zero and no error messages.

After this job, the SYSLIB concatenation needs to be updated in the target and distribution zones:
- Add the SASMMAC1, SASMSAM1, and AASMMAC1 libraries to the SYSLIB concatenation in the target zone.
- Add the AASMMAC1 library to the SYSLIB concatenation in the distribution zone.

Step 8: Perform SMP/E apply

There are two methods of applying High Level Assembler:
- The interactive capability of SMP/E described in "Method 1: Apply using SMP/E dialogs."
- The batch procedure described in "Method 2: Apply using a batch job."

To receive the full benefit of the SMP/E Causer SYSMOD Summary Report, do *not* bypass the following on the APPLY CHECK: PRE, ID, REQ, and IFREQ. This is because the SMP/E root cause analysis identifies the cause only of **ERRORS** and not of **WARNINGS** (SYSMODs that are bypassed are treated as warnings, not errors, by SMP/E).

Method 1: Apply using SMP/E dialogs

1. Select the Command Generation option from the SMP/E Primary Option menu.

   You will be asked to supply some of the information above as you progress through subsequent command generation panels for APPLY.
2. Perform an APPLY CHECK before the APPLY. Examine the output from the APPLY CHECK run.

   Expected Return Codes and Messages: This should issue a return code of zero and no error messages.
3. If it shows no conflict, rerun the APPLY with CHECK=NO.

   Expected Return Codes and Messages: This should issue a return code of zero and no error messages.

Method 2: Apply using a batch job

1. Perform an Apply with Check

   Use the sample job ASMWAPLY to apply High Level Assembler.

   The installation job ASMWAPLY invokes SMP/E to apply High Level Assembler. ASMWAPLY performs an APPLY with the CHECK option to check for possible conflicts in the target zone.

   Edit and submit sample job ASMWAPLY to perform an SMP/E APPLY CHECK. Consult the instructions in the sample job for more information.

Expected Return Codes and Messages: This job should issue a return code of zero and no error messages.

2. Perform an Apply

   Edit sample job ASMWAPLY to remove the CHECK operand and resubmit.

   Note: The GROUPEXTEND operand indicates that SMP/E accept all requisite SYSMODs. The requisite SYSMODS might be applicable to other functions.

   Expected Return Codes and Messages: This job should issue a return code of zero and no error messages.

Step 9: Run the installation verification program

A sample job, ASMWIVP, is provided to verify that the product has installed correctly. This job assembles the source statements from member ASMASAMP. The assembly of ASMASAMP verifies that the product is installed and functions correctly.

Edit and submit sample job ASMWIVP. Consult the instructions in the sample job for more information.

Expected Return Codes and Messages: This job should issue a return code of zero and no error messages.

Step 10: Perform SMP/E accept

To permanently install High Level Assembler, use SMP/E ACCEPT processing. You should accept High Level Assembler *before* you apply any user modifications or install any maintenance because the SMP/E RESTORE command restores High Level Assembler only to the level of the last version you accept.

To receive the full benefit of the SMP/E Causer SYSMOD Summary Report, do *not* bypass the following on the ACCEPT CHECK: PRE, ID, REQ, and IFREQ. This is because the SMP/E root cause analysis identifies the cause only of **ERRORS** and not of **WARNINGS** (SYSMODs that are bypassed are treated as warnings, not errors, by SMP/E).

There are two methods of accepting High Level Assembler:
- The interactive capability of SMP/E described in "Method 1: Accept using SMP/E dialogs."
- The batch procedure described in "Method 2: Accept using a batch job."

Method 1: Accept using SMP/E dialogs

1. Select the Command Generation option from the SMP/E Primary Option menu.

   Supply the information requested as you progress through subsequent command generation panels for ACCEPT.

2. Perform an ACCEPT CHECK before running the ACCEPT command. Examine the output from the ACCEPT CHECK run.

   Expected Return Codes and Messages: This should issue a return code of zero and no error messages.

3. Perform an ACCEPT.

   If the ACCEPT CHECK run shows no conflict, rerun the ACCEPT with CHECK=NO.

   Expected Return Codes and Messages: This should issue a return code of zero and no error messages.

Method 2: Accept using a batch job

1. Perform an ACCEPT with CHECK

The job ASMWACPT performs an ACCEPT with the CHECK option to check for possible conflicts in the distribution zone.

Edit and submit sample job ASMWACPT to perform an SMP/E ACCEPT CHECK for High Level Assembler. Consult the instructions in the sample job for more information.

Expected Return Codes and Messages:. This job should issue a return code of zero and no error messages.

2. Perform an ACCEPT

   When this job is successful, rerun this job without the CHECK option.

   Before using SMP/E to load new distribution libraries, it is recommended that you set the ACCJCLIN indicator in the distribution zone. This will cause entries produced from JCLIN to be saved in the distribution zone whenever a SYSMOD containing inline JCLIN is ACCEPTed. For more information about the ACCJCLIN indicator, see the description of inline JCLIN in *SMP/E Commands*.

   Note: The GROUPEXTEND operand indicates that SMP/E accept all requisite SYSMODs. The requisite SYSMODS might be applicable to other functions.

   Expected Return Codes and Messages: This job should issue a return code of zero and no error messages.

Chapter 4. Customizing High Level Assembler on z/OS

You can customize, or modify, High Level Assembler only after installing the product. This chapter includes:
- SMP/E Considerations
- Customization Checklist
- Customization Steps

SMP/E considerations

The following SMP/E information should be considered prior to commencing any customizing task.

1. Install High Level Assembler as described in Chapter 3, "Installing High Level Assembler on z/OS," on page 17.
2. Apply the USERMODs to the target libraries, *but do not accept them into the distribution libraries.*
3. Use SMP/E RESTORE to remove a USERMOD before you apply service to the modules it changes.
4. Reapply the USERMOD after successful installation of High Level Assembler service.
5. Sample jobs are provided to assist you in customizing High Level Assembler. The sample jobs can be found in the ASM.SASMSAM1 library. The sample jobs are:

   ASMAASM
   Sample SMP/E job to make HLASM the default Assembler utility

   ASMAOPTS
   A sample SMP/E USERMOD job to change default options

   ASMASTD
   Sample job to copy the HLASM procedures to your PROCLIB

   ASMAIEV
   Sample SMP/E USERMOD job to create IEV90 alias

   In the sample SMP/E jobs provided, the name of the SMP/E CSI is `#globalcsi`. The global zone name in the SMP/E CSI is `GLOBAL`. The distribution zone name is `#dzone`. The target zone name is `#tzone`. The sample jobs should be updated to reflect the CSI and zone names used at your installation. Refer to the instructions in the sample jobs for other changes you may need to make.

Customization checklist

Table 20 lists the steps and associated jobs to customize High Level Assembler. The remaining sections in this chapter describe each step. You can use Table 20 as a checklist.

Table 20. Summary of steps for customizing High Level Assembler

Step	Description	Customizing Job	Page
__ 1	Change the SMP/E assembler utility defaults.	ASMAASM	26
__ 2	Customize user exits.	_	27
__ 3	Change default options and DDNAMES.	ASMAOPTS	26
__ 4	Customize the High Level Assembler Procedures.	ASMASTD	28
__ 5	Place High Level Assembler into link pack area.	_	31
__ 6	Assembler H Migration (optional). Complete this step only as an aid to migrating JCL procedures to reference ASMA90 as the assembler program name.	ASMAIEV	31

Customization steps

Depending on your requirements not all of the following steps need to be performed. For example, if you do not wish to change the supplied default options and DDNAMES, you can bypass that step.

Step 1: Change SMP/E to use High Level Assembler

The default assembler utility used by SMP/E is High Level Assembler. If this default has been changed at your site, you can reinstate it using the sample job, ASMAASM, which is provided to help you do this.

Depending on how your SMP/E system is configured, you must specify ASMA90 as the name of the program to be called in the SMP/E Utility entry.

You can also consider changing other existing OPTIONS entries. For example, you can alter the ASM subentry of each OPTIONS entry to point to the new UTILITY entry of High Level Assembler.

If your OPTIONS entry specifies High Level Assembler as the assembler in SMP/E, then you must make ASM.SASMMOD1 APF-authorized.

Edit and submit job ASMAASM to change the OPTIONS entry. Consult the instructions in the sample job for more information. The sample job is shown in Figure 3.

```
//ASMAASM  JOB  <JOB CARD PARAMETERS>
//*
//ALTER    EXEC PGM=GIMSMP
//SMPCSI   DD DSN=#globalcsi,
//            DISP=SHR
//SMPOUT   DD SYSOUT=*
//SMPRPT   DD SYSOUT=*
//SMPLIST  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPCNTL  DD *
 SET BDY(GLOBAL).              /* INIT GLOBAL ZONE CSI              */
 UCLIN .
   ADD OPTIONS(GBLOPT)         /* ADD AN OPTIONS ENTRY              */
       ASM(HLASM)              /* SPECIFY NAME OF ASSEMBLER         */
       .
   ADD UTILITY(HLASM)          /* ADD UTILITY ENTRY FOR ASSEMBLER   */
                               /* DEFINED IN ABOVE OPTIONS ENTRY    */
       NAME(ASMA90)            /* NAME OF PROGRAM TO BE INVOKED     */
       RC(4)                   /* RETURN CODE THRESHOLD             */
       PRINT(SYSPRINT)         /* DDNAME FOR SYSPRINT OUTPUT        */
                               /* DEFAULT PARAMETERS                */
       PARM(NOOBJ,DECK,XREF(SHORT,UNREFS))                          */
       .
 ENDUCL.
 LIST ALLZONES.                /* LIST ZONE INFORMATION             */
/*
```

Figure 3. Change SMP/E to use High Level Assembler

Step 2: Customize user exits

For information about customizing the user exits, see the *HLASM Programmer's Guide*. Exits are provided for:
- Source exit
- Listing exit
- ADATA exit

Sample source for these exits is provided in the ASM.SASMSAM1 target library. These samples are:

ASMAXINV
Sample source exit.

ASMAXPRT
Sample listing exit.

To support the ADATA exit the following are provided:

ASMAXFSK
Sample skeleton SYSADATA filter routine

ASMAXFMT
Sample filter management table

ASMAXFLU
Sample filter module to dump ADATA records

ASMAXADT
Sample ADATA exit to load filter routine

ASMAXADR
Sample exit to reformat ADATA records from new to old format

ASMAXADC
Sample exit to control ADATA record output

Step 3: Change default options and DDNAMES

The job ASMAOPTS supplies a USERMOD to SMP/E to create a modified options module ASMADOPT.

Before you run ASMAOPTS:

1. Refer to "General rules for coding the ASMADOPT ASSEMBLE file" on page 63 for more information about how to enter the options into the sample job.
2. Review the worksheets in "Choices to make now" on page 10, and enter these options and DDNAMES values into the sample job ASMAOPTS.
3. Edit and submit the sample job, ASMAOPTS, to change the default options. Consult the instructions in the sample job for more information. This job is shown in Figure 4 on page 28.

```
//ASMAOPTS JOB  <JOB CARD PARAMETERS>
//*
//USERMOD  EXEC PGM=GIMSMP
//SMPCSI   DD DSN=#globalcsi,
//            DISP=SHR
//SMPOUT   DD SYSOUT=*
//SMPRPT   DD SYSOUT=*
//SMPLIST  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD  DSN=#hlq.SASMMAC1,
//             DISP=SHR
//STEPLIB  DD  DSN=#hlq.SASMMOD1,
//             DISP=SHR
//SMPPTFIN DD  *
++USERMOD(ML00001) REWORK(#date).          1
++VER(Z038) FMID(HMQ4160).                 2
++SRC(ASMADOPT) DISTLIB(AASMSAM1).
    ASMAOPT ADATA=YES                      3
    ASMADD  UT1=SYSUT2                     4
    END
/*
//SMPCNTL  DD  *
   SET BDY(GLOBAL).
   RECEIVE S(ML00001) SYSMODS.
   SET BDY(#tzone).
   APPLY S(ML00001) ASSEM.                 5
/*
```

Figure 4. Changing default options and DDNAMES

To change the default options and DDNAMES:

1. In area **1** use the REWORK option if you change the USERMOD frequently.

   Specify REWORK if the USERMOD name will be reused every time and you do not prefer to use the REJECT command before receiving the USERMOD.

   Every time you re-apply the USERMOD, increment the value of REWORK using the form of *yyyyddd*, where *yyyy* is the year (for example, 2008) and *ddd* is the day of the year (for example, 036 is the 5th February). For more information refer to SMP/E Reference.

   If you do not increment the REWORK option, then on subsequent APPLY commands the REDO operand is required on the SMP/E APPLY statement.
2. In area **2** the ++VER may require a PRE(UKxxxxx) parameter on the statement if any maintenance has already been applied to the product. Refer to SMP/E Reference and SMP/E Commands for more information.
3. In area **3** specify new default assembler values for options. The sample shows *ADATA=YES* being selected as an example. Only include values for options that you want to change.
4. In area **4** specify new default assembler values for DDNAMES. The sample shows *UT1=SYSUT2* being selected as an example. Only include values for DDNAMEs that you want to change.
5. In area **5** you might want to use the COMPRESS option on the APPLY statement to minimize storage used for some or all of the target libraries.
6. Ensure that all PTF service which affects part ASMADOPT is ACCEPTed before applying USERMOD ML00001 to avoid regressing service on that part.

For the USERMOD, do not perform any ACCEPT processing.

Step 4: Customize the High Level Assembler procedures

Four standard procedures are supplied with High Level Assembler. They are:

ASMAC
: Assembles only

ASMACL
: Assembles and link-edits a program

ASMACLG
: Assembles, link-edits a program, and runs the program

ASMACG
: Assembles and uses the loader to run the program

They are distributed in the target library, ASM.SASMSAM1.

If you have installed the High Level Assembler load modules into your own library, you will need to add a STEPLIB DD statement for your library, to the procedures.

You may decide to move the High Level Assembler procedures to your system procedure library. A sample job, ASMASTD is supplied to help you do this.

Edit and submit the job ASMASTD to move the procedures. Consult the instructions in the sample job for information about changes you may need to make. Figure 5 on page 30 shows this sample job.

This job moves the procedures into SYS1.PROCLIB; if necessary change this to match your system procedure library.

As an alternative to moving these procedures to a JES- defined procedure library, consider using a JCLLIB JCL statement

The names of the supplied procedures changed after High Level Assembler Release 1. Therefore the original Release 1 procedures are supplied as aliases in the target library ASM.SASMSAM1. If you also want to move the aliases to SYS1.PROCLIB or other JES defined procedure libraries then include another SELECT statement in the ASMASTD job. For example:

```
SELECT MEMBER=HLASMC,HLASMCG,HLASMCLG,HLASMCL
```

In some circumstances you may receive the following message when running these procedures:

```
IEF686I DDNAME REFERRED TO ON DDNAME KEYWORD
        IN PRIOR STEP WAS NOT RESOLVED
```

This warning message indicates that there were no linkage editor control statements. You supply these statements following a SYSIN DDNAME statement. If you did not supply any linkage-editor control statements, this message can be ignored.

Figure 6 on page 30, Figure 7 on page 30, Figure 8 on page 31, and Figure 9 on page 31 show these procedures as shipped with High Level Assembler.

```
//ASMASTD  JOB  <JOB CARD PARAMETERS>
//*
//         EXEC PGM=IEBCOPY
//SASMSAM1 DD  DSN=#hlq.SASMSAM1,
//             DISP=SHR
//OUT      DD  DSN=SYS1.PROCLIB,DISP=OLD
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
         COPY INDD=SASMSAM1,OUTDD=OUT
         SELECT MEMBER=ASMAC,ASMACG,ASMACLG,ASMACL
/*
```

Figure 5. Copying procedures from sample library to procedure library

Supplied procedures

The following figures display the contents of each supplied procedure.

```
//ASMAC    PROC
//*
//C        EXEC PGM=ASMA90
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD  DSN=&&SYSUT1,SPACE=(4096,(120,120),,,ROUND),
//             UNIT=SYSALLDA,DCB=BUFNO=1
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&OBJ,SPACE=(3040,(40,40),,,ROUND),
//             UNIT=SYSALLDA,DISP=(MOD,PASS),
//             DCB=(BLKSIZE=3040,LRECL=80,RECFM=FB,BUFNO=1)
```

Figure 6. ASMAC procedure

```
//ASMACL   PROC
//*
//C        EXEC PGM=ASMA90
//SYSLIB   DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD  DSN=&&SYSUT1,SPACE=(4096,(120,120),,,ROUND),
//             UNIT=SYSALLDA,DCB=BUFNO=1
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&OBJ,SPACE=(3040,(40,40),,,ROUND),
//             UNIT=SYSALLDA,DISP=(MOD,PASS),
//             DCB=(BLKSIZE=3040,LRECL=80,RECFM=FB,BUFNO=1)
//L        EXEC PGM=HEWL,PARM='MAP,LET,LIST,NCAL',COND=(8,LT,C)
//SYSLIN   DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DISP=(,PASS),UNIT=SYSALLDA,SPACE=(CYL,(1,1,1)),
//             DSN=&&GOSET(GO)
//SYSUT1   DD  DSN=&&SYSUT1,SPACE=(1024,(120,120),,,ROUND),
//             UNIT=SYSALLDA,DCB=BUFNO=1
//SYSPRINT DD  SYSOUT=*
```

Figure 7. ASMACL procedure

```
//ASMACLG   PROC
//*
//C         EXEC PGM=ASMA90
//SYSLIB    DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1    DD  DSN=&&SYSUT1,SPACE=(4096,(120,120),,,ROUND),
//              UNIT=SYSALLDA,DCB=BUFNO=1
//SYSPRINT DD  SYSOUT=*
//SYSLIN    DD  DSN=&&OBJ,SPACE=(3040,(40,40),,,ROUND),
//              UNIT=SYSALLDA,DISP=(MOD,PASS),
//              DCB=(BLKSIZE=3040,LRECL=80,RECFM=FB,BUFNO=1)
//L         EXEC PGM=HEWL,PARM='MAP,LET,LIST',COND=(8,LT,C)
//SYSLIN    DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//          DD  DDNAME=SYSIN
//SYSLMOD   DD  DISP=(,PASS),UNIT=SYSALLDA,SPACE=(CYL,(1,1,1)),
//              DSN=&&GOSET(GO)
//SYSUT1    DD  DSN=&&SYSUT1,SPACE=(1024,(120,120),,,ROUND),
//              UNIT=SYSALLDA,DCB=BUFNO=1
//SYSPRINT DD  SYSOUT=*
//G         EXEC PGM=*.L.SYSLMOD,COND=((8,LT,C),(8,LT,L))
```

Figure 8. ASMACLG procedure

```
//ASMACG    PROC
//*
//C         EXEC PGM=ASMA90
//SYSLIB    DD  DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1    DD  DSN=&&SYSUT1,SPACE=(4096,(120,120),,,ROUND),
//              UNIT=SYSALLDA,DCB=BUFNO=1
//SYSPRINT DD  SYSOUT=*
//SYSLIN    DD  DSN=&&OBJ,SPACE=(3040,(40,40),,,ROUND),
//              UNIT=SYSALLDA,DISP=(MOD,PASS),
//              DCB=(BLKSIZE=3040,LRECL=80,RECFM=FB,BUFNO=1)
//G         EXEC PGM=LOADER,PARM='MAP,LET,PRINT',COND=(8,LT,C)
//SYSLIN    DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//          DD  DDNAME=SYSIN
//SYSLOUT   DD  SYSOUT=*
```

Figure 9. ASMACG procedure

Step 5: Place High Level Assembler into link pack area

Table 17 on page 14 lists the modules distributed with High Level Assembler Release 6, and their attributes. You can use this table to determine which, if any, modules you want to move to the LPA.

These modules are installed into the target library, ASM.SASMMOD1. The modules may be copied to a linklist library, an LPA library or a private library depending upon your requirements.

Step 6: Assembler H migration (optional)

To simplify migration of existing assembler procedures, the IEV90 module can be given as an alias of ASMA90. A sample usermod, ASMAIEV is supplied in the ASM.SASMSAM1 library to do this. Only run this usermod as an aid to migrating references from IEV90 to ASMA90. When all references have been changed, remove the usermod.

Edit and submit job ASMAIEV to create an alias of ASMA90. Consult the instructions in the sample job for information about changes you may need to make. ASMAIEV is shown in Figure 10 on page 32.

```
//ASMAIEV  JOB  <JOB CARD PARAMETERS>
//*
//USERMOD  EXEC PGM=GIMSMP
//SMPCSI   DD DSN=#globalcsi,
//            DISP=SHR
//SMPOUT   DD SYSOUT=*
//SMPRPT   DD SYSOUT=*
//SMPLIST  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD  DSN=#hlq.SASMMAC1,
//             DISP=SHR
//STEPLIB  DD  DSN=#hlq.SASMMOD1,
//             DISP=SHR
//SMPPTFIN DD  DATA,DLM=QX
++USERMOD(ML00009) REWORK(#date).       1
++VER(Z038) FMID(HMQ4160).
++JCLIN.
//S2      EXEC LINKS,
//             PARM='XREF,MAP,NCAL,RENT,REUS,REFR,AMOD=31,RMOD=ANY',
//             UNIT='',SER=,N=,NAME=SASMMOD1,P1='',
//             MOD=,P2='',OBJ=,CLASS=
//AASMMOD1 DD  DISP=SHR,DSN=#hlq.AASMMOD1
//SYSLIN   DD  *
 ORDER ASMA90
 INCLUDE AASMMOD1(ASMACPR)
 INCLUDE AASMMOD1(ASMA90)
 ALIAS IEV90
 ENTRY ASMA90
 NAME  ASMA90(R) RC=0
/*
++MOD(ASMA90) LKLIB(SASMMOD1).
QX
//SMPCNTL  DD  *
   SET BDY(GLOBAL).
   RECEIVE S(ML00009) SYSMODS.
   SET BDY(#tzone).
   APPLY S(ML00009).                    2
/*
```

Figure 10. Create an IEV90 alias

When running this USERMOD consider the following:

1. Ensure that all PTF service which affects part ASMA90 is ACCEPTed before applying USERMOD ML00009 to avoid regressing service to that part.
2. In area **1** use the REWORK option if you change the USERMOD frequently.

   Specify REWORK if the USERMOD name will be reused every time and you do not prefer to use the REJECT command before receiving the USERMOD.

   Every time you re-apply the USERMOD, increment the value of REWORK using the form of *yyyyddd*, where *yyyy* is the year (for example, 2008) and *ddd* is the day of the year (for example, 036 is the 5th February). For more information refer to SMP/E Reference.

   If you do not increment the REWORK option, then on subsequent APPLY commands the REDO operand is required on the SMP/E APPLY statement.
3. In area **2** you might want to use the COMPRESS option on the APPLY statement to minimize storage used for some or all of the target libraries.

For the USERMOD, do not perform any ACCEPT processing.

Chapter 5. Maintaining High Level Assembler on z/OS

This chapter describes how to re-install or remove High Level Assembler and how to apply service updates. To use the maintenance procedures effectively, you should have already installed High Level Assembler and any required products.

Re-installing High Level Assembler

The action required here depends on the circumstance. If you want to re-install High Level Assembler Release 6, and you did not use the SMP/E ACCEPT command then use an SMP/E APPLY REDO command. However, if you did use the SMP/E ACCEPT command, then the product should be deleted before installing again. For more information refer to "Removing High Level Assembler" on page 34.

Applying service updates

You might need to apply maintenance or service updates to High Level Assembler periodically.

What you receive

If you report a problem with High Level Assembler to your IBM Support Center, you will receive a tape containing one or more APARs or PTFs that have been created to solve your problem.

You might also receive a list of pre-requisite APARs or PTFs, which should have been applied to your system before applying the current service. These prerequisite APARs or PTFs, might relate to High Level Assembler or any other licensed product you have installed, including z/OS.

To help you understand the service process, the following overview familiarizes you with applying service for High Level Assembler.

Checklist for applying service

Table 21 lists the steps and associated SMP/E commands for installing corrective service on High Level Assembler. You can use Table 21 as a checklist.

Table 21. Summary of steps for installing service on High Level Assembler

Step	Description	SMP/E Command	Page
__ 1	Prepare to install service.		33
__ 2	Receive service.	RECEIVE	34
__ 3	Accept previously applied service (optional).	ACCEPT	34
__ 4	Apply service.	APPLY	34
__ 5	Test service.		34
__ 6	Accept service.	ACCEPT	34

Step 1. Prepare to install service

Before you start applying service:

1. Create a backup copy of the current High Level Assembler. Save this copy of High Level Assembler until you have completed installing the service and you are confident that the service runs correctly.
2. Research each service tape through the IBM Support Center for any errors and additional information. Note all errors on the tape that were reported by APARs and apply the applicable fixes.

Step 2. Receive the service

Receive the service using SMP/E RECEIVE command. This can be done from the SMP/E dialogs in ISPF or using a batch job.

Step 3. Accept applied service (optional)

Accept any service you applied earlier but did not accept, if you are satisfied that the earlier service is not causing problems in your installation. This can be done from the SMP/E dialogs in ISPF or using a batch job. Accepting the earlier service allows you to use the SMP/E RESTORE command to return to your current level if you encounter a problem with the service you are currently applying. This can be done from the SMP/E dialogs in ISPF or using a batch job.

Step 4. Apply the service

Apply the service using SMP/E APPLY command. You should use the SMP/E APPLY command with the CHECK operand first. Check the output; if it shows no conflict, rerun the APPLY without the CHECK option. This can be done from the SMP/E dialogs in ISPF or using a batch job.

Do not apply the documented USERMODs until PTF service has been ACCEPTed. This is to avoid regressing service to the affected parts.

Step 5. Test the service

Thoroughly test your updated High Level Assembler. Run the installation verification program to ensure that the product functions properly. You can use the job, ASMWIVP, from the target library ASM.SASMSAM1, to do this. Do not accept a service update until you are confident that it runs correctly.

In the event of a serious problem, you can restore the backup copy of High Level Assembler.

Step 6. Accept the service

Accept the service using SMP/E ACCEPT command. You should use the SMP/E ACCEPT command with the CHECK operand first. Check the output; if it shows no conflict, rerun the ACCEPT without the CHECK option. This can be done from the SMP/E dialogs in ISPF or using a batch job.

Removing High Level Assembler

To delete High Level Assembler, you must:

- Make sure no other products depend on it.
- Use a dummy function SYSMOD to delete it.
- Receive, apply, and accept the dummy function, and run the UCLIN to delete the SYSMOD entries for the deleted function and the dummy function.

Edit and submit job ASMADEL0 to delete High Level Assembler. Consult the instructions in the sample job for more information.

Expected Return Codes and Messages: You receive message `GIM39701W` because the dummy function SYSMOD has no elements. The SMP/E RECEIVE command returns a return code of 4. If any USERMODs have been applied then the SMP/E APPLY command issues a `GIM44502W` message indicating USERMOD changes will be lost with a return code of 4. Both these warning messages can be ignored.

The target and distribution libraries can now be deleted. They are shown in Table 7 on page 4 and Table 8 on page 5.

Reporting a problem with High Level Assembler

If you have a problem with High Level Assembler, you can first check to see whether the resolution to your problem is already documented, at:

`http://www.ibm.com/software/awdtools/hlasm/support.html`

If you cannot find an answer, report any difficulties with this product to your IBM Support Center. If an APAR is required, the Support Center will provide the address to which any needed documentation can be sent.

To assist with reporting any difficulties refer to the diagnostic process as shown in Chapter 20, “Isolating the problem,” on page 115.

Table 22 identifies the component ID (COMP ID) for High Level Assembler for z/OS.

Table 22. Component IDs

FMID	COMP ID	Component Name	REL
HMQ4160	569623400	MVS HIGH LEVEL ASM	160

Obtaining service information

Preventive Service Planning (PSP) information is continually updated as fixes are made available for problems. Check with your IBM Support Center or use either Information/Access or SoftwareXcel Extended to see whether there is additional PSP information you need. To obtain this information, specify the following UPGRADE and SUBSET values: HLASM160 and HMQ4160.

Chapter 6. Planning for installing High Level Assembler on z/VM

This section contains the following planning information to help you properly install High Level Assembler on z/VM:

- Worksheet
- What you receive with High Level Assembler
- Identifying required and optional software
- VMSES/E considerations
- Verifying that you have enough DASD storage
- Deciding to install in a saved segment
- Checking service updates
- Publications useful during installation

Worksheet: Planning for installing High Level Assembler on z/VM

Before you begin the installation you should:

1. Determine which of the following you are installing High Level Assembler as:

   Part of a z/VM System Delivery Offering

   By itself

   If installing High Level Assembler by itself, determine the product parts to be installed:

   COMPID *569623400*

   Feature number *5872*

   See "What you receive with High Level Assembler."
2. Verify that required software (and optional software, if appropriate) is at the level needed. See "What you need to install High Level Assembler" on page 39.
3. Ensure you are familiar with VMSES/E. See "VMSES/E considerations" on page 39.
4. Verify that adequate storage is available:

   Minidisks

   Shared File System

   See "DASD storage required" on page 39.
5. Decide whether to install in a saved segment.

   See "Planning to install in a saved segment" on page 40.
6. Obtain latest service updates needed. See "Program support" on page 41.

What you receive with High Level Assembler

You receive the following when you order High Level Assembler for z/VM:

COMPIDs	Feature Numbers	System Name
569623400	5872	z/VM

Distribution media

High Level Assembler is available through the z/VM SDO on 3590 and 3592 tape cartridge. You can also receive this program electronically by ordering it through the z/VM SDO using IBM ShopzSeries. For more information about IBM ShopzSeries go to `www.ibm.com/software/ShopzSeries.`

The tape cartridge or electronic envelope contains all the programs and data needed for installation.

Basic material

Table 23 describes the cartridge. Table 24 describes the file content of the program cartridge.

Table 23. Basic material: program cartridge

Medium	Feature Number	Physical Volume	External Label Identification	VOLSER
3480 cart.	5872	1	HLASM V1R6 for VM	unlabeled

Table 24. Program cartridge: file content

File	Content
1	Tape Header
2	Tape Header
3	Product Header
4	Product Memo
5	Service Apply Lists
6	PTFPARTS
7	High Level Assembler Service
8	High Level Assembler Service summary
9	High Level Assembler Base code
10	High Level Assembler Sample and customization files
11	High Level Assembler Executable product code

Optional material

There are no optional materials for High Level Assembler.

Cumulative service tape

You might receive an additional tape containing cumulative service with your order. The PTFs on this tape have not yet been incorporated into this release.

If you received this product as part of a z/VM System Delivery Offering, PTFs that have not been incorporated into this release are provided on the tape. A separate cumulative service tape is *not* provided.

Program publications and softcopy

This section identifies the basic and optional publications for High Level Assembler.

- *HLASM Licensed Program Specifications* GC26-4944
- *HLASM Installation and Customization Guide* SC26-3494
- *HLASM Language Reference* SC26-4940
- *HLASM Programmer's Guide* SC26-4941
- *HLASM General Information* GC26-4943

For a list of books for related products, see “Bibliography” on page 191.

What you need to install High Level Assembler

The following sections identify the system requirements for installing High Level Assembler.

Operating system requirements

High Level Assembler Release 6 supports the following z/VM operating systems, and subsequent releases and versions:

- z/VM Version 5

Other program product requirements

No other products are required for High Level Assembler.

VMSES/E considerations

This section describes items that should be considered before you install High Level Assembler

- VMSES/E is required to install and service this product.
- The VMSES/E system-level software inventory and other dependent files reside on the MAINT 51D disk. If multiple users install and maintain licensed products on your system there might be a problem getting the necessary write access to MAINT's 51D disk. If you find that there is contention for write access to the 51D disk, you can eliminate it by converting the software inventory from minidisk to the Shared File System (SFS). See *z/VM: VMSES/E Introduction and Reference* for information about how to make this change.
- You no longer install and service High Level Assembler strictly using the MAINT user ID, but using a new user ID P696234J. IBM recommends that you use this userid, however, you can change the userid name if you wish. If you change this name, you must create a PPF override. Generally, this section of this installation and customization guide assumes that you have used the IBM-recommended userid, P696234J.

  If you plan to change the userid, it might be easier to do so during Step 1, substep 6 on page 48, while you are installing the product.
- If you plan to install to minidisks, you should be aware that the saved segment must be built from the installation user ID P696234J. If you want to build the segment from a common user ID, install High Level Assembler using the SFS system, so that the common user ID can access High Level Assembler code when building segments.

DASD storage required

Before installing you need to understand the user ID and DASD storage requirements. Some important points to consider are:

- The installation user ID and minidisks or SFS are defined in “Step 2: Allocate resources for installing High Level Assembler” on page 49. Table 25 on page 40 gives an estimate of the storage required to install High Level Assembler for z/VM.
- P696234J is the default user ID and can be changed. If you choose to change the name you need to create a Product Parameter File (PPF) override to change the name. This can be done in Step 1, substep 6 on page 48.
- If you choose to install High Level Assembler on a common user ID, the default minidisk addresses for High Level Assembler might already be defined. If any of the default minidisks required by High Level Assembler are already in use, create an override to change the minidisks for High Level Assembler so they are unique.

Table 25. DASD storage requirements for target minidisks

Minidisk Owner (user ID)	Default Address	Storage in Cylinders		SFS 4K Blocks	Usage
		DASD	CYLS		Default SFS Directory Name
P696234J	2B2	3390	10	1800	Contains all the base code shipped with High Level Assembler.
					VMSYS: P696234J.HLASM.OBJECT
P696234J	2C2	3390	4	720	Contains sample files, user local modifications and user exits for High Level Assembler.
					VMSYS: P696234J.HLASM.LOCAL
P696234J	2D2	3390	20	1800	Contains serviced files.
					VMSYS: P696234J.HLASM.DELTA
P696234J	2A6	3390	1	180	Contains AUX files and software inventory files that represents your test service level of High Level Assembler.
					VMSYS: P696234J.HLASM.APPLYALT
P696234J	2A2	3390	1	180	Contains AUX files and software inventory files that represents the service level of High Level Assembler that is currently in production.
					VMSYS: P696234J.HLASM.APPLYPROD
P696234J	29E	3390	6	1080	Test build disk. This code will be copied to a production disk, (for example, MAINT 19E) so the production disk will also require this amount of free space.
					VMSYS: P696234J.HLASM.TBUILD
P696234J	191	3390	5	900	P696234J 191 minidisk.
					VMSYS:P696234J
Note: Cylinder values defined in this table are based on 4K block size.					

If you plan to use SFS directories, then calculate the number of 4K blocks required from Table 25.

Planning to install in a saved segment

Defining frequently used data (such as licensed programs) as saved segments provides several advantages:

- Several users can access the same physical storage, therefore use of real storage is minimized.
- Using saved segments decreases the input and output rate and the DASD paging space requirements, thus improving the performance of the virtual machine.
- Saved segments attached to a virtual machine can reside above its defined virtual storage.
- In the install process a physical saved segment is defined and its logical saved segment loaded.

Segment planning should be done at the system level. A VMSES/E tool, VMFSGMAP provides system-level management support functions.

In “Step 5: Define and build High Level Assembler saved segment (optional)” on page 54, the process of defining and building a saved segment is shown in detail. If you are unfamiliar with this process, refer to *z/VM: Saved Segments Planning and Administration*.

Program support

This section describes the IBM support available for High Level Assembler.

Program services

Contact your IBM representative for specific information about available program services.

Preventive Service Planning

Before installing High Level Assembler, you should review the current Preventive Service Planning (PSP) information.

PSP Buckets are identified by UPGRADEs, which specify product levels, and SUBSETs, which specify the FMIDs for a product level. The UPGRADE and SUBSET values for High Level Assembler for z/VM are:

Table 26. PSP upgrade and subset ID

UPGRADE	SUBSET	Description
HLASM160	HLASMVM360	HLASM VM

Statement of support procedures

Report any difficulties you have using this program to your IBM Support Center. If an APAR is required, the Support Center will provide the address to which any needed documentation can be sent.

Table 27 identifies the component ID (COMPID) for High Level Assembler for z/VM.

Table 27. Component IDs

COMPID	Component Name	RETAINRelease
569623400	VM HIGH LEVEL ASM	360

Program and service level information

This section identifies the program and any relevant service levels of High Level Assembler. The program level refers to the APAR fixes incorporated into the program. The service level refers to the PTFs integrated. Information about the cumulative service tape is also provided.

Program level information

A list of APAR fixes against previous releases of High Level Assembler that have been incorporated into this release is shown in Appendix C, "High Level Assembler Service," on page 183.

Service level information

No PTFs against this release of High Level Assembler have been incorporated into the product tape.

Publications useful during installation

The publications listed in Table 28 may be useful during the installation of High Level Assembler for z/VM.

Table 28. z/VM publications

Publication Title	Form Number
z/VM: VMSES/E Introduction and Reference	GC24-6243
z/VM: Service Guide	GC24-6247

Table 28. z/VM publications (continued)

Publication Title	**Form Number**
z/VM: CMS Commands and Utilities Reference	SC24-6166
z/VM: CMS File Pool Planning, Administration, and Operation	SC24-6167
z/VM: CP Planning and Administration	SC24-6178
z/VM: Saved Segments Planning and Administration	SC24-6229
z/VM: Other Components Messages and Codes	GC24-6207
z/VM: CMS and REXX/VM Messages and Codes	GC24-6161
z/VM: CP System Messages and Codes	GC24-6177

Chapter 7. Planning for customizing High Level Assembler on z/VM

This section provides information for planning for the customization of High Level Assembler on z/VM. It includes:
- Deciding whether and what to customize
- Planning to customize the default option and DDNAMES
- Planning to customize the user exits

Deciding whether and what to customize

You need to consider whether the IBM-supplied values that come with High Level Assembler suit the needs of your site. These values control such features as:
- Selecting multicultural support
- Assembler Options
- Default file names (DDNAMES)
- User exits

Make sure that High Level Assembler serves the needs of the application programmers at your site. Confer with them while you evaluate the customization options for High Level Assembler, particularly those concerning High Level Assembler options that are also available to the application programmers. Doing so will ensure that the modifications you make best support the application programs being developed at your site.

The information in this chapter helps you plan your customization. See Chapter 9, “Customizing High Level Assembler on z/VM,” on page 63 for the actual customization procedure.

Selecting multicultural support

When customizing High Level Assembler, you can choose which language you want to use for diagnostic messages. Languages available are English (mixed case or upper case), German, Spanish, and Japanese. You specify the language by means of the LANGUAGE option. See page “LANGUAGE” on page 154 for information about the LANGUAGE option.

The following combinations are possible:

English Uppercase
Diagnostic messages and listing headings printed in uppercase English

English Mixed Case
Diagnostic messages and listing headings printed in mixed case English

German
Diagnostic messages in German and listing headings printed in mixed case English.

Japanese
Diagnostic messages in Kanji and listing headings printed in uppercase English.

Spanish
Diagnostic messages in Spanish and listing headings printed in mixed case English.

Planning to customize High Level Assembler options and DDNAMES

High Level Assembler can be customized with a large range of options and DDNAMES for users. When customized they become the default options and DDNAMES.

Why do it

The High Level Assembler options and DDNAMES should be reviewed to assess the required defaults for your site. Worksheets are provided for planning purposes.

Choices to make now

The following worksheets help you plan and code the options and DDNAMES appropriate for your site. To complete the worksheets, fill in the **Enter Selection** column.

Table 29. Worksheet options

Option	Enter Selection	IBM-SuppliedDefault	Description
ADATA	__________	NO	page 143
ADEXIT	__________	no exit specified	page 144
ALIGN	__________	YES	page 144
ALIGNWARN	__________	YES	page 145
BATCH	__________	YES	page 145
CODEPAGE	__________	047C	page 146
COMPAT	__________	NO	page 146
CONTWARN	__________	YES	page 147
DBCS	__________	NO	page 148
DECK	__________	NO	page 148
DELETE	__________	no options deleted	page 149
DSECT	__________	NO	page 150
DXREF	__________	YES	page 150
ESD	__________	YES	page 151
EXLITW	__________	YES	page 151
FLAG	__________	0	page 151
FOLD	__________	NO	page 152
IMPLENWARN	__________	NO	page 153
INEXIT	__________	no exit specified	page 153
INFO	__________	NO	page 154
LANGUAGE	__________	EN	page 154
LIBEXIT	__________	no exit specified	page 155
LIBMAC	__________	NO	page 155
LIMIT	__________	NO	page 156
LINECOUNT	__________	60	page 156
LIST	__________	YES	page 157
MACHINE	__________	no default, see OPTABLE	page 158
MACHINELIST	__________	NO	page 159
MAP	__________	YES	page 159
MXREF	__________	SOURCE	page 160
OBJECT	__________	YES	page 160
OBJEXIT	__________	no exit specified	page 161
OPTABLE	__________	UNI	page 161

Table 29. Worksheet options (continued)

Option	Enter Selection	IBM-SuppliedDefault	Description
OPTABLELIST	__________	NO	page 163
PAGE0WARN	__________	NO	page 163
PCONTROL	__________	NO	page 164
PESTOP	__________	NO	page 165
PROFILE	__________	NO	page 166
PROFMEM	__________	ASMAPROF	page 166
PRTEXIT	__________	no exit specified	page 167
PUSHWARN	__________	YES	page 167
RA2	__________	NO	page 167
RECORDINFO	__________	YES	page 168
RENT	__________	NO	page 168
RLD	__________	YES	page 169
RXREF	__________	YES	page 169
SECTALGN	__________	8	page 169
SIZE	__________	MAX	page 170
STORAGE	__________	nothing specified	page 171
SUBSTRWARN	__________	NO	page 171
SUPRWARN	__________	NO	page 172
SYSPARMV	__________	none specified	page 172
TERM	__________	NO	page 173
TEST	__________	NO	page 173
THREAD	__________	YES	page 173
TRANSLATE	__________	NO	page 174
TRMEXIT	__________	no exit specified	page 174
TYPECHECK	__________	(MAGNITUDE, REGISTER)	page 175
USING0WARN	__________	YES	page 175
WARN	__________	15	page 176
WORKFILE	__________	NO	page 176
XREF	__________	(SHORT,UNREFS)	page 178

Table 30. Worksheet: DDNAMES

DDNAME	EnterSelection	IBM-SuppliedDefault	Description
ADATA	__________	SYSADATA	page 179
IN	__________	SYSIN	page 179
LIB	__________	SYSLIB	page 180
LIN	__________	SYSLIN	page 180
OPT	__________	ASMAOPT	page 180
PRINT	__________	SYSPRINT	page 181
PUNCH	__________	SYSPUNCH	page 181

Table 30. Worksheet: DDNAMES (continued)

DDNAME	EnterSelection	IBM-SuppliedDefault	Description
TERM	__________	SYSTERM	page 181
UT1	__________	SYSUT1	page 182

Planning to customize the user exits

Several exits are provided with the High Level Assembler, they are described in the *HLASM Programmer's Guide*. The exit types are:
- Source exit
- Listing exit
- ADATA exit

In order to help you customize your exits, sample source is provided, as follows:

ASMAXINV
  Sample exit to read variable length records

ASMAXPRT
  Sample listing exit

To support the ADATA exit the following are provided:

ASMAXFSK
  Sample skeleton SYSADATA filter routine

ASMAXFMT
  Sample filter management table

ASMAXFLU
  Sample filter module to dump ADATA records

ASMAXADT
  Sample ADATA exit to load filter routine

ASMAXADR
  Sample exit to reformat ADATA records from new to old format

ASMAXADC
  Sample exit to control ADATA record output

Note: If User Exits from a previous release are being used, these exits may need to be reassembled for this release.

Chapter 8. Installing High Level Assembler on z/VM

This chapter describes the installation method and the step-by-step procedures you use to install and activate the functions of High Level Assembler.

Overview of installation

You install this release of High Level Assembler by using VMSES/E commands.

Checklist for installing High Level Assembler

Table 31 lists the steps and associated VMSES/E commands for installing High Level Assembler. The remaining sections in this chapter describe each step. You can use Table 31 as a checklist.

Table 31. Summary of steps for installing High Level Assembler

Step	Description	VMSES/E Command	Page
__ 1	Prepare to install High Level Assembler.	VMFINS	47
__ 2	Allocate resources for installing High Level Assembler.		49
__ 3	Install High Level Assembler.	VMFINS	51
__ 4	Verify the installation in a test environment.	VMFINS	53
__ 5	Define and build High Level Assembler saved segment (optional).	VMFSGMAP VMFBLD	54
__ 6	Place High Level Assembler into production.	VMFCOPY	59

Step 1: Prepare to install High Level Assembler

Carry out the following preparatory tasks:

1. Log on as the High Level Assembler installation planner, typically the MAINT user ID. IBM recommends that you use P696234J as the product installation user ID.

   Normally, the MAINT user ID is used for the planning and setup for High Level Assembler, then the P696234J user ID is used to install the product.

   However, you can log on as any user ID that has read access to MAINT's 5E5 minidisk and write access to the MAINT 51D minidisk.
2. Make the installation media available:

   If you are installing from tape, attach the tape to the user ID at virtual address 181. Using virtual address 181 is required by the VMFINS EXEC.

   If you have received High Level Assembler as an Envelope file, ensure the Envelope file exists on the A-disk, or any disk accessed as C.
3. Establish read access to VMSES/E, using the following commands:

Command	Explanation

`link MAINT 5e5 5e5 rr`
`access 5e5 b`
	Commands to access VMSES/E code.

4. Establish write access to the software inventory disk.

Command	Explanation
link MAINT 51d 51d mr **access 51d d**	Commands to acquire write access to software inventory disk.

If another user already has the software inventory disk linked in write mode (R/W), you are not allowed to link to that disk in write mode. The other user must relink to the software inventory disk in read-only mode, then you can try the above procedure again.

The VMSES/E system-level software inventory and other dependent files reside on the MAINT 51D disk.

You cannot proceed with the installation procedure until you have write access to the software inventory disk.

5. Load the High Level Assembler product control files onto the software inventory disk.

   If you are installing from tape:

Command
vmfins install info (nomemo

If you are installing from an Envelope file:

Command
vmfins install info (nomemo env *filename*

This command will:
- Load the Memo-to-Users file
- Load the product control files including the Product Parameter File (PPF) and the PRODPART files
- Create VMFINS PRODLIST on your A-disk. The VMFINS PRODLIST contains a list of products on the installation tape.

Options:

filename
    The name of your Envelope file, for example, P696234J.

nomemo
    Loads the memo but does not issue a prompt to send it to the system printer. Specify the **memo** option if you want to be prompted to print the memo.

6. Obtain the resource planning information.

   If you are installing from tape:

Command
vmfins install ppf P696234J {HLASM|HLASMSFS} (plan nomemo

If you are installing from an Envelope file:

Command
vmfins install ppf P696234J {HLASM|HLASMSFS} (plan nomemo env *filename*

You are prompted to change the installation defaults with message VMFINS2601R. Refer to the notes below for the changes you can make.

This command creates P696234J PLANINFO file on your A disk.
- Use `HLASM` if High Level Assembler is to be installed on a minidisk.
- Use `HLASMSFS` if High Level Assembler is to be installed in SFS.

Note: This command *does not* load High Level Assembler.

Options:

filename
The name of your Envelope file, for example, P696234J.

nomemo
Loads the memo from the tape but does not issue a prompt to send it to the system printer. Specify the **memo** option if you want to be prompted to print the memo.

plan VMFINS performs requisite checking, plans system resources, and provides an opportunity to override the defaults in the product parameter file.

Notes:

a. You can override:
   - The name of the product parameter file
   - The default user IDs
   - Minidisk/directory definitions
b. If you change the PPF name, a default user ID, or other parameters via a PPF override, you must use your changed values instead of those indicated (when appropriate), throughout the rest of the installation instructions, as well as the instructions for servicing High Level Assembler. For example, you must specify your PPF override file name instead of P696234J for certain VMSES/E commands.
c. If you are not familiar with creating PPF overrides using VMFINS, refer to *z/VM: VMSES/E Introduction and Reference* before you continue. Also see Appendix D, "Create Product Parameter File (PPF) override (z/VM)," on page 185.

7. Review the install message log file.

Command	Explanation
`vmfview install`	View the install message log.

Review the install message log ($VMFINS $MSGLOG). If necessary, correct any problems before going on. For information about specific error messages, see *z/VM: Other Components Messages and Codes*.

Step 2: Allocate resources for installing High Level Assembler

Use the information from the P696234J PLANINFO file to allocate storage resources for your installation.

In the planning chapter, you decided whether to install on minidisk or shared file system (SFS). See "DASD storage required" on page 39 for the relevant storage requirements.

Common instructions for allocating storage resources

1. Obtain the user directory from the P696234J PLANINFO file.
   - User directory entries contain all of the links and privilege classes necessary for the P696234J user ID.
   - Use the directory entry in PLANINFO as a model for input to your system directory.

   The user directory entry resides at the end of the PLANINFO file, at the end of the resource requirements section.
2. Add the P696234J directory to the system directory.

   Change the password for P696234J from xxxxx to a valid password in accordance with the security guidelines at your site.

Minidisk installers: Go to "If installing on a minidisk" on page 50.

SFS installers: Go to "If installing on a shared file system."

If installing on a minidisk

After obtaining the user directory from the P696234J PLANINFO file:
- Add the MDISK statements to the directory entry for P696234J.
- Use Table 25 on page 40 in Chapter 6, "Planning for installing High Level Assembler on z/VM," on page 37 to obtain the minidisk requirements.
- Place the new directory online using VM/Directory Maintenance (DIRMAINT) or an equivalent CP directory maintenance method.

Go to "Step 3: Install High Level Assembler" on page 51.

If installing on a shared file system

After completing the common steps above, proceed as follows:
1. If you want to use an SFS directory as the work space for the P696234J top directory, include the following in the P696234J directory entry. This change requires the directory to be rebuilt and placed online.

Directory Statement	Explanation
`ipl cms parm filepool VMSYS`	This directory statement allows automatic access to the P696234J's top directory as file mode A.

2. Place the new directory online using VM/Directory Maintenance (DIRMAINT) or an equivalent CP directory maintenance method.
3. Use the number of 4K blocks calculated from Table 25 on page 40.
   - If you are installing all the default High Level Assembler SFS directories, the block requirements are summarized in the table.
   - If you are selectively installing SFS directories, calculate the number of 4K blocks your installation requires by adding up the storage required by each of the directories you plan to install.
4. From a user ID that is an administrator for the VMSYS filepool, issue the following command:

Command	Explanation
`enroll user P696234J vmsys: (blocks` *`blocks`*	This command enrolls the user P696234J in the VMSYS filepool. `VMSYS` is the default filepool.

   Options:

   blocks The number of 4K blocks that you calculated previously.

   This makes available a top directory of VMSYS:P696234J. to the user.

5. Compare your storage needs to the amount of storage available in the filepool.

Command	Explanation
`query filepool status vmsys:`	Command to get a list of directories in the filepool and the number of free blocks. `VMSYS` is the default filepool ID.

   If the number of free blocks is smaller than the number of blocks you need to install High Level Assembler (which you calculated in step 3, above), you need to add space to the filepool.

   For instructions on adding space to a filepool, see *z/VM: CMS File Pool Planning, Administration, and Operation*.

6. Create the necessary `vmsys:P696234J.HLASM` subdirectories used in the P696234J PLANINFO file.

Command	Explanation
set filepool vmsys: **create directory** *dirid*	Commands to create subdirectories.

Options:

dirid The name of the SFS directory you are creating.

Examples of the command above include:

```
create directory vmsys:P696234J.HLASM
create directory vmsys:P696234J.HLASM.object
⋮
```

A complete list of default High Level Assembler SFS directories is provided in Table 25 on page 40. For information about the CREATE DIRECTORY command, see *z/VM: CMS Commands and Utilities Reference*.

There is no need to create the top directory (VMSYS:P696234J.) as this has been implicitly created by the ENROLL command.

7. Give the MAINT user ID READ authority to the general-use test build directory, using the GRANT AUTHORITY command.

Command	Explanation
grant auth vmsys:P696234J.HLASM.TBUILD to MAINT (read newread	The GRANT command permits copying files to MAINT's 19E disk.

Options:

read Gives the user **read** authority on a file or directory.

newread
Indicates that users automatically receive **read** authority for any new files added to the directory.

For more information about the GRANT AUTHORITY command, see *z/VM: CMS Commands and Utilities Reference*.

Go to “Step 3: Install High Level Assembler”

Step 3: Install High Level Assembler

1. Log on to the installation user ID, P696234J.
2. Format the minidisks

   If you are installing to minidisks rather than SFS directories, then format these disks now. For a list of disk addresses, refer to Table 25 on page 40.
3. Create or modify the PROFILE EXEC with access commands for MAINT's 5E5 and 51D disks. To do this, insert the following statements into the PROFILE EXEC:

Command	Explanation
'access 5e5 b' **'access 51d d'**	Statements to access the 5E5 and 51D disks.

4. Execute the profile to access MAINT's minidisks.

Command	Explanation
profile	Command to execute profile.

5. Verify that you have write access to the software inventory disk.

Command	Explanation
`q 51d`	Command to query access status of the software inventory disk.

If the disk is not in read write mode (R/W) then see Step 1, substep 4 on page 47.

6. Make the installation media available:

If you have received High Level Assembler as a product tape, then attach the tape to the user ID at virtual address 181. Using virtual address 181 is required by the VMFINS EXEC.

If you have received High Level Assembler as an Envelope file, ensure the Envelope file exists on the A-disk.

7. Install High Level Assembler product from tape or from the Envelope file.

You might be prompted for additional information during VMFINS INSTALL processing, depending on your installation environment. If you are unsure how to respond to a prompt, refer to the appropriate sections in *z/VM: VMSES/E Introduction and Reference*.

If you are installing from tape:

Command
`vmfins install ppf P696234J {HLASM|HLASMSFS} (nomemo nolink`

If you are installing from an Envelope file:

Command
`vmfins install ppf P696234J {HLASM|HLASMSFS} (nomemo nolink env` *filename*

This command installs High Level Assembler.
- Use HLASM if High Level Assembler is installed on a minidisk.
- Use HLASMSFS if High Level Assembler is installed in SFS.

You are prompted to create an override for High Level Assembler when this command executes. Refer to the message text for your response.

Options:

filename
The name of your Envelope file name, for example, P696234J.

nomemo
Loads the memo from the tape but does not issue a prompt to send it to the system printer. Specify the **memo** option if you want to be prompted to print the memo.

nolink
Indicates that VMFINS is not to link to the appropriate minidisks, only access them if they are not already accessed.

Note:

a. You can override:
   - The name of the product parameter file
   - The default user IDs
   - Minidisk/directory definitions
b. If you change the PPF name, a default user ID, or other parameters via a PPF override, you must use your changed values instead of those indicated (when appropriate), throughout the rest of the

installation instructions, as well as the instructions for servicing High Level Assembler. For example, you must specify your PPF override file name instead of P696234J for certain VMSES/E commands.

c. If you are not familiar with creating PPF overrides using VMFINS, refer to *z/VM: VMSES/E Introduction and Reference* before you continue. Also see Appendix D, "Create Product Parameter File (PPF) override (z/VM)," on page 185.

8. Review the install message log ($VMFINS $MSGLOG).

Command	Explanation
`vmfview install`	View the install message log.

Correct any problems before you go on. For information about specific error messages, see *z/VM: Other Components Messages and Codes*.

Step 4: Verify the installation in a test environment

If you need to change any of the default options or DDNAMES then refer to Chapter 9, "Customizing High Level Assembler on z/VM," on page 63. When you have completed any customization, return to this step.

When you issue the command to update the build status table for High Level Assembler, VMSES/E also executes the verification EXEC, V5696234.

1. Update Build Status Table and run verification EXEC.

Command	Explanation
`vmfins build ppf P696234J {HLASM│HLASMSFS} (serviced nolink`	Update VM SYSBLDS software inventory file for High Level Assembler Use `HLASM` if High Level Assembler is installed on a minidisk. Use `HLASMSFS` if High Level Assembler is installed in SFS.

Options:

serviced
: Identifies build requirements and builds those objects flagged as **serviced** in the service-level build status table.

nolink
: Indicates that VMFINS is not to link to the appropriate minidisks, only to access them if they are not already accessed.

A verification test is run automatically to ensure that High Level Assembler was installed successfully. Figure 11 shows sample screen output from the verification EXEC.

```
:
VMFINB2173I Executing verification exec V5696234
****************************************************
*** Verification assembly completed successfully
***
*****************************************************
VMFINS2760I VMFINS processing completed successfully
```

Figure 11. Sample install verification output

2. Review the install message log ($VMFINS $MSGLOG).

Command	Explanation
`vmfview install`	View the install message log.

Correct any problems before you go on. For information about specific error messages, see *z/VM: Other Components Messages and Codes*.

Step 5: Define and build High Level Assembler saved segment (optional)

If you do not plan to place High Level Assembler into a saved segment then bypass this step and continue with "Step 6: Put High Level Assembler into production" on page 59.

A logical saved segment provides several advantages such as saving on real storage, and decreased input/output; also a saved segment can reside above the defined virtual storage of a virtual machine. To build a saved segment you must:

1. Define the physical saved segment using the segment mapping tool VMFSGMAP.
2. Use VMFBLD to build it.

For more information about using VMSES/E for saved segments, see *z/VM: Saved Segments Planning and Administration*. Also see the SEGGEN command in *z/VM: CMS Commands and Utilities Reference*.

Defining and building the High Level Assembler saved segment should be performed from the installation user ID. If you move any segments that are currently defined on your system, you must ensure that they are rebuilt from the user ID that maintains them.

1. Logon to the installation user ID P696234J.
2. Establish read access to VMSES/E.

Command	Explanation

`link MAINT 5e5 5e5 rr`
`access 5e5 b`
	Commands to access VMSES/E.

3. Establish write access to the software inventory disk.

Command	Explanation

`link MAINT 51d 51d mr`
`access 51d d`
	Commands to access software inventory files.

If another user already has the software inventory disk linked in write mode (R/W), you are not allowed to link to that disk in write mode. The other user must relink to the software inventory disk in read-only mode, then you can try the above procedure again.

The MAINT 51D disk is where the VMSES/E system-level software inventory and other dependent files reside.

You cannot proceed with the installation procedure until you have write access to the software inventory disk.

4. Add High Level Assembler segment object definitions to the SEGBLIST EXC00000 build list.

All the modules that are to reside in the logical saved segment have an RMODE=ANY and a AMODE=31. Therefore, the physical saved segment that is about to be defined can reside above 16MB.

Command	Explanation
`vmfsgmap segbld esasegs segblist`	This command displays a panel for making segment updates.

This command displays a panel for making segment updates, shown in Figure 12.

```
                              VMFSGMAP - Segment Map                  More: +
                                                               Lines 1 to nn of nn

             000-MB          001-MB          002-MB          003-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M CMS      SYS W-W-------------1...............2...............3...............
M GCS      SYS W---------------1...............2...............3...............

             004-MB          005-MB          006-MB          007-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  CMSPIPES DCS 4...............5...............6...............RRRRRR----------
M GCS      SYS RRRRRRRNNNNNNNNNNNNNNNNNNNNNNNNNN6...............7...............

             008-MB          009-MB          00A-MB          00B-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  DOSBAM   SPA 8...............9...............A...............====------------
  CMSBAM   MEM 8...............9...............A...............RRRR............
  CMSDOS   MEM 8...............9...............A...............R...............
  CMSVMLIB DCS RRRRRRRRRRRRRRRR9...............A...............B...............

             00C-MB          00D-MB          00E-MB          00F-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  HELPINST DCS RRRRRRRRRRRRRRRRD...............E...............F...............
 F1=Help     F2=Chk Obj   F3=Exit      F4=Chg Obj   F5=File       F6=Save
 F7=Bkwd     F8=Fwd       F9=Retrieve  F10=Add Obj  F11=Del Obj   F12=Cancel
====>__
```

Figure 12. Segment map panel example

Browse through this segment. If the ASMAPSEG segment is encountered it should be removed by placing the cursor under the ASMAPSEG name and pressing PF11. A successful deletion will give message

```
VMFSMD2046I Segment ASMAPSEG has been deleted
```

5. Go to Add Segment Definition panel by pressing PF10.

   Press **F10** to take you from the Segment Map panel to the Add Segment Definition panel. See Figure 13 to see the Add Segment Definition panel displays.

```
                              Add Segment Definition
                                                  Lines 1 to nn of nn

OBJNAME....:  ????????
DEFPARMS...:
SPACE......:
TYPE.......:  SEG
OBJDESC....:
OBJINFO....:
GT_16MB....:  NO
DISKS......:
SEGREQ.....:
PRODID.....:
BLDPARMS...:  UNKNOWN




 F1=Help     F2=Get Obj   F3=Exit      F4=Add Line  F5=Map        F6=Chk MEM
 F7=Bkwd     F8=Fwd       F9=Retrieve  F10=Seginfo  F11=Adj MEM   F12=Cancel
====>
```

Figure 13. Add segment definition panel example

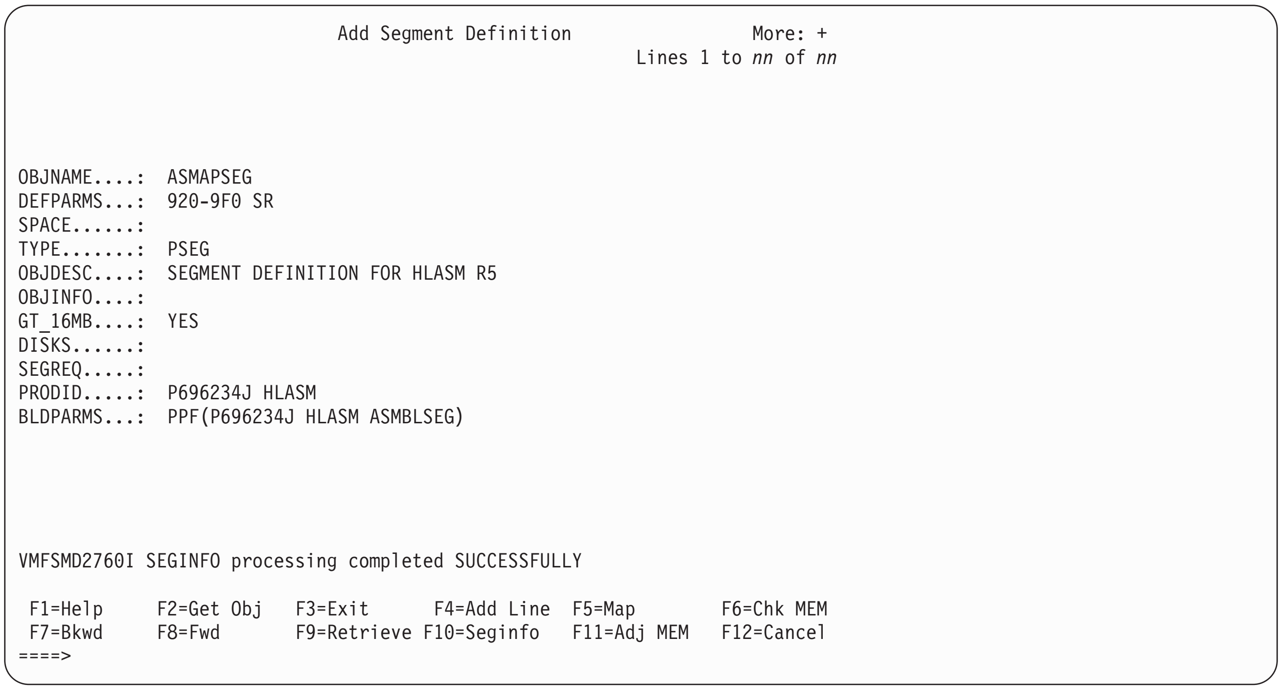
6. Obtain the High Level Assembler segment definitions from the PRODPART file. Enter the following highlighted values into the 'Add Segment Definition Panel'.

Command	Explanation
OBJNAME....: **ASMAPSEG** PRODID.....: **P696234J {HLASM│HLASMSFS}**	Use HLASM for building a segment from a minidisk. Use HLASMSFS for building a segment from SFS directories.

F10	**F10** obtains the High Level Assembler segment information from the P696234J PRODPART file.

See Figure 14 for the refreshed Add Segment definition panel that is displayed.

If you have created a PPF override file then replace the PPF name on the *BLDPARMS*. In this example, replace P696234J with your PPF override name.



```
                              Add Segment Definition                More: +
                                                    Lines 1 to nn of nn




OBJNAME....:  ASMAPSEG
DEFPARMS...:  920-9F0 SR
SPACE......:
TYPE.......:  PSEG
OBJDESC....:  SEGMENT DEFINITION FOR HLASM R5
OBJINFO....:
GT_16MB....:  YES
DISKS......:
SEGREQ.....:
PRODID.....:  P696234J HLASM
BLDPARMS...:  PPF(P696234J HLASM ASMBLSEG)





VMFSMD2760I SEGINFO processing completed SUCCESSFULLY

 F1=Help     F2=Get Obj   F3=Exit      F4=Add Line  F5=Map        F6=Chk MEM
 F7=Bkwd     F8=Fwd       F9=Retrieve F10=Seginfo   F11=Adj MEM   F12=Cancel
====>
```

Figure 14. Add segment definition panel showing the new segment

7. Go back to the Segment Map panel.

Command	Explanation
F5	**F5** returns you to the Segment Map panel.

See Figure 15 on page 57 for the refreshed Segment Map panel that displays.

```
                          VMFSGMAP - Segment Map            More: -
                                           Lines nn to nn of nn




               000-MB          001-MB          002-MB          003-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
M CMS      SYS W-W-------------1...............2...............3...............
M GCS      SYS W---------------1...............2...............3...............

               004-MB          005-MB          006-MB          007-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  CMSPIPES DCS 4...............5...............6...............RRRRRR----------
M GCS      SYS RRRRRRNNNNNNNNNNNNNNNNNNNNNNNNNN6...............7...............

               008-MB          009-MB          00A-MB          00B-MB
  Name     Typ 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
  DOSBAM   SPA 8...............9...............A...............====------------
  CMSBAM   MEM 8...............9...............A...............RRRR............
  CMSDOS   MEM 8...............9...............A...............R...............
  CMSVMLIB DCS RRRRRRRRRRRRRRRR9...............A...............B...............
  DOSINST  DCS 8...............R---------------A...............B...............
P ASMAPSEG DCS 8...............--RRRRRRRRRRRR--A...............B...............

               00C-MB          00D-MB          00E-MB          00F-MB
 F1=Help     F2=Chk Obj   F3=Exit      F4=Chg Obj   F5=File       F6=Save
 F7=Bkwd     F8=Fwd       F9=Retrieve  F10=Add Obj  F11=Del Obj   F12=Cancel
====>
```

Figure 15. Segment map panel with added segments

8. Save the new information and exit from the Segment Map panel.

   Press **F5** to save all changed information and exit the map panel.
9. Prepare to build the segment.
   a. Set storage.

Command	Explanation
def stor 24M	This CP command defines the virtual machine storage to be greater than where the segment is loaded. If defining a saved segment above 16MB then the virtual machine storage should be above 16MB. In general the storage should be large enough to hold the segment. If you are using SFS in this user ID, even if only for the A disk, and you are placing the ASMAPSEG segment below 16MB at 920-9F0 then use a 'DEF STOR 24M' command. This will allow CMS control blocks not to interfere with the storage area to be used by the High Level Assembler saved segment.

   b. IPL the virtual machine **without** accessing the profile.

   If IPLing with minidisks:

Command	Explanation
`ipl cms parm clear nosprof instseg no mtseg no` *** DO NOT press ENTER at the VM READ***	

Command	Explanation
	This command clears your virtual machine and bypasses the execution of the system profile (SYSPROF EXEC) and the loading of the installation saved segment (CMSINST).
access (noprof	Bypass the execution of the PROFILE EXEC.

If IPLing with SFS:

Command	Explanation
`ipl cms parm clear nosprof instseg no mtseg no filepool vmsys` *** DO NOT press ENTER at the VM READ***	This command clears your virtual machine and bypasses the execution of the system profile (SYSPROF EXEC) and the loading of the installation saved segment (CMSINST). This assumes `VMSYS` is the default filepool.
access (noprof	Bypass the execution of the PROFILE EXEC.

c. Access disks

Command	Explanation
`access 5e5 b`	Access the VMSES/E code.
`link MAINT 51d 51d mr` `access 51d d`	Establish write access to the software inventory disk.

10. Issue VMFBLD command to build the High Level Assembler segment.

Command	Explanation
`vmfbld ppf segbld esasegs segblist ASMAPSEG (serviced`	

Options:

serviced
Identifies build requirements and builds those objects flagged as **serviced** in the service-level build status table.

11. Update the SYSTEM SEGID file on the CMS system disk.

The system segment identification file (SYSTEM SEGID) must reside on the CMS system disk (normally file mode S) and is named SYSTEM SEGID so that it is available to CMS at initialization time. This allows CMS to recognize the logical saved segment name.

The file must also be copied to the test CMS system disk to prevent back-levelling during application of service.

The following functions can only be performed by authorized user IDs.

a. Release disk for use by MAINT user ID.

Command	Explanation
`rel d (det`	Release the software inventory disk.

b. Log on to MAINT user ID.

c. Move SYSTEM SEGID file

Command	Explanation
`acc 190 t`	Access the production CMS system disk (normally MAINT 190).
`acc 490 v`	Access the test CMS system disk (normally MAINT 490).
`acc 51d d`	Access the software inventory disk.
`copyfile system segid d system segid t2 (replace olddate`	Copy file to production CMS system disk.
`copyfile system segid d system segid v2 (replace olddate`	Copy file to test CMS system disk.

Options:

replace
Causes the output file to replace an existing file with the same file identifier.

olddate
Uses the date and time on each input file as the date and time of the last update of each corresponding output file.

The CMS system is not resaved here, it is done after moving High Level Assembler to MAINT's 19E disk in "Step 6: Put High Level Assembler into production."

Step 6: Put High Level Assembler into production

You now have the choice of:
- "Putting High Level Assembler onto the Y disk"
- "Putting High Level Assembler onto a disk other than the Y disk" on page 61.

This completes the installation process for a disk other than the Y disk.

Putting High Level Assembler onto the Y disk

This section describes how to put High Level Assembler into production on the CMS Y disk (MAINT's 19E disk). If you plan to use another disk, go to "Putting High Level Assembler onto a disk other than the Y disk" on page 61.

1. Log on to MAINT.
2. Link to High Level Assembler code and access the Y disk.

   Choose which access is required depending on whether you have used a minidisk or SFS.

   For minidisk access:

Command	Explanation

`link P696234J 29e 29e rr`
`access 29e e`
	Access High Level Assembler code on a minidisk.

`access 19e f`	Access CMS's Y disk.

For SFS access:

Command	Explanation
`access vmsys:P696234J.HLASM.TBUILD e`	Access High Level Assembler code in an SFS directory.
`access 19e f`	Access CMS's Y disk.

3. Check if a previous High Level Assembler release is present.

   If the Y disk contains a previous release, then you should remove it. Migration is covered in substep 5, below. High Level Assembler code can be identified by modules with a prefix of *ASMA* and the *HLASM* module (Release 1 only).
4. Move High Level Assembler to the Y disk.

   If you have created a saved segment for High Level Assembler, then all modules except the following four reside in the saved segment. Therefore you only need to copy these four modules with the VMFCOPY command:
   - ASMAHL
   - ASMA90
   - ASMADOPT
   - ASMAINFO

   If you have not created a saved segment then all the High Level Assembler modules should be placed onto the Y disk using the VMFCOPY command.

   You should copy the ASMAMAC MACLIB, whether or not you have created a saved segment for High Level Assembler.

   If you have created a saved segment for High Level Assembler, use the following VMFCOPY commands:

Command

```
vmfcopy asmahl   module e = = f2 (prodid P696234J%HLASM replace olddate
vmfcopy asma90   module e = = f2 (prodid P696234J%HLASM replace olddate
vmfcopy asmadopt module e = = f2 (prodid P696234J%HLASM replace olddate
vmfcopy asmainfo module e = = f2 (prodid P696234J%HLASM replace olddate
vmfcopy * maclib e = = f2 (prodid P696234J%HLASM replace olddate
```

   If you have not created a saved segment for High Level Assembler, use the following VMFCOPY commands:

Command

```
vmfcopy * module e = = f2 (prodid P696234J%HLASM replace olddate
vmfcopy * maclib e = = f2 (prodid P696234J%HLASM replace olddate
```

   Options:

   replace
      Causes the output file to replace an existing file with the same file identifier.

   olddate
      Uses the date and time on each input file as the date and time of the last update of each corresponding output file.

   prodid
      Causes the VMSES PARTCAT file on the 19E disk to be updated.
5. Migration from previous Assemblers.

   If Release 1 of High Level Assembler has been removed then the ASMAHL MODULE must be copied to the Y disk as HLASM MODULE. This will allow z/VM service to function correctly.

   If you want to migrate from previous assemblers, you should consider the following:
   - HLASM MODULE in High Level Assembler Release 1 can be replaced by ASMAHL MODULE in this release of High Level Assembler.
   - HASM MODULE in Assembler H can be replaced by ASMAHL MODULE in this release of High Level Assembler. However, ASMAHL MODULE does not support the *NUM* and *STMT* options.

- IEV90 MODULE in Assembler H can be replaced by the ASMA90 MODULE in this release of High Level Assembler. If you replace IEV90 MODULE with the ASMA90 MODULE, then you should also copy ASMAHL MODULE to HASM MODULE.

You can then copy them to the Y disk, using the following commands.

Command
`vmfcopy asmahl module e HLASM = f2 (prodid P696234J%HLASM replace olddate`
`vmfcopy asmahl module e HASM = f2 (prodid P696234J%HLASM replace olddate`
`vmfcopy asma90 module e IEV90 = f2 (prodid P696234J%HLASM replace olddate`

Options:

replace
Causes the output file to replace an existing file with the same file identifier.

olddate
Uses the date and time on each input file as the date and time of the last update of each corresponding output file.

6. Move User Exits (Optional)

   User exits are supplied as sample assembler source on the LOCAL disk. If these exits are to be made generally available, they can be moved (ASMAX... ASSEMBLE) to the MAINT 19E disk, or the LOCAL disk can be made available to the users.

7. Resave the CMS saved system

   This updates the shared Y-STAT (the saved Y disk file directory). If you have created a saved segment, it also updates the S-STAT (the saved S disk file directory) to reflect the updated SYSTEM SEGID file.

   a. Define named saved system

      This creates a named saved system for CMS.

Command	Explanation
`vmfsetup zvm cms`	On z/VM, allows access to the SAMPNSS EXEC which resides on MAINT's 193 disk.
`sampnss cms`	Create named saved system for CMS.

   b. Resave the CMS system

Command	Explanation
`ipl 190 clear parm savesys cms`	Resave CMS saved system to reflect the change to the S-STAT (the saved S disk file directory).

This completes the installation process for the Y disk.

Putting High Level Assembler onto a disk other than the Y disk

If you plan to use a disk other than the Y disk, log on on to the user ID that owns the disk you plan to use. Then follow the instructions in “Putting High Level Assembler onto the Y disk” on page 59, steps 2 to 7, replacing references to the 19E disk or the Y disk with the address or name of the disk you want to use.

Chapter 9. Customizing High Level Assembler on z/VM

You can customize, or modify, High Level Assembler only after installing the product. This chapter includes:
- Changing the option and DDNAME defaults.
- Changing the saved segment name.
- Customizing the user exits.

Changing option and DDNAMES defaults

You can customize the default options and DDNAMES for High Level Assembler for z/VM. Refer to "Planning to customize High Level Assembler options and DDNAMES" on page 43 to identify what default options or DDNAMES you intend to change.

You use VMSES/E to create a local modification that changes the default options and DDNAMES. You modify ASMADOPT ASSEMBLE on the LOCAL disk.

Appendix E, "Local modification procedures (z/VM)," on page 187 shows how a local modification procedure is carried out for this product. For more information see *z/VM: Service Guide*.

Before modifying ASMADOPT, read the following sections.

General rules for coding the ASMADOPT ASSEMBLE file

- Code in columns 2 through 71.
- Do not put a comma in front of the first option in your macro.
- Begin any continuation lines in column 16 and place a non-blank character in column 72 of the previous line. You can break the coding after any comma.
- Place an END statement after the macro instruction lines.
- Include only those options or DDNAMES that you want to change. High Level Assembler uses the IBM-supplied defaults for the unchanged options and DDNAMES.
- For a description of full syntax for the High Level Assembler options, see Appendix A, "High Level Assembler Options," on page 143.
- For a description of full syntax for the High Level Assembler DDNAMES, see Appendix B, "High Level Assembler DDNAMES (z/OS and CMS)," on page 179.

Example of Modifying Option and DDNAME Values

ASMADOPT looks likes this:

```
         ASMAOPT
         END
```

Figure 16. ASMADOPT example

For example, you might make the following changes, shown in Figure 17 on page 64.
- Change the value for the XREF option to SHORT, and for the PESTOP option to YES.
- Change the value for DDNAME UT1 to SYSUT2.

```
ASMADOPT        CSECT
                ASMAOPT XREF=SHORT,PESTOP=YES
                ASMADD  UT1=SYSUT2
                END
```

Figure 17. ASMADOPT example modified

Changing the saved segment name

The default name for the physical saved segment is ASMAPSEG. There is no restriction on the name of the physical saved segment. For testing purposes, High Level Assembler can reside in more than one physical saved segment. To switch between them, you use the SEGMENT ASSIGN command.

The name of the logical saved segment is ASMA93 and cannot be changed. You use the SEGMENT LOAD command to load the logical saved segment.

To provide another physical saved segment for testing purposes, follow the instructions in “Step 5: Define and build High Level Assembler saved segment (optional)” on page 54 with the following changes:

1. Change the name of the physical saved segment on the 'Add Segment Definition' panel within the VMFSGMAP tool as shown in Figure 14 on page 56.
2. Create a *PSEG* file. This could be done by copying the *ASMAPSEG PSEG* file on the test build disk and renaming it to the new name.
3. Change the physical segment name on the *VMFBLD* command in substep 10 on page 58 from ASMAPSEG to the new name.

Customizing the user exits

For information about customizing the user exits, see the *HLASM Programmer's Guide*. Exits are provided for:

- Source exit
- Listing exit
- ADATA exit

Sample source for these exits is distributed when you install High Level Assembler. These samples are:

ASMAXINV
Sample source exit.

ASMAXPRT
Sample listing exit.

To support the ADATA exit the following are provided:

ASMAXFSK
Sample skeleton SYSADATA filter routine

ASMAXFMT
Sample filter management table

ASMAXFLU
Sample filter module to dump ADATA records

ASMAXADT
Sample ADATA exit to load filter routine

ASMAXADR
Sample exit to reformat ADATA records from new to old format

ASMAXADC
: Sample exit to control ADATA record output

Chapter 10. Maintaining High Level Assembler on z/VM

This chapter describes how to re-install, or remove High Level Assembler and how to apply service updates. To use the maintenance procedures, you must have already installed High Level Assembler and any required products.

To become more familiar with service using VMSES/E, refer to *z/VM: VMSES/E Introduction and Reference*. This manual also contains the command syntax for the VMSES/E commands listed in the procedure.

Each step of the service instructions must be followed; do not skip any step unless otherwise directed. All instructions showing accessing of disks assume default minidisk addresses. If different minidisk addresses are used, or if using a shared-file system, change the instructions appropriately.

Re-installing High Level Assembler

You should delete the product and commence the installation process from the beginning. To delete the product refer to "Removing High Level Assembler" on page 77. When the product is deleted, start the installation from "Step 1: Prepare to install High Level Assembler" on page 47.

Applying service updates

This section describes how to apply maintenance or service updates to High Level Assembler.

What you receive

If you report a problem with High Level Assembler to your IBM Support Center, you will receive a tape containing one or more APARs or PTFs which solve your problem.

You might also receive a list of pre-requisite APARs or PTFs, which should have been applied to your system before applying the current service. These pre-requisite APARs or PTFs might relate to High Level Assembler or any other licensed product you have installed, including z/VM.

The following overview familiarizes you with some of the aspects of applying service for High Level Assembler.

Checklist for applying service

Table 32 lists the steps and associated VMSES/E commands for installing corrective service on High Level Assembler. You can use Table 32 as a checklist.

Table 32. Summary of steps for installing service on High Level Assembler

Step	Description	VMSES/E Command	Page
__ 1	Prepare to install service.		68
__ 2	Clear the alternate APPLY disk. Doing this will allow you to remove the new service easily later if necessary.	VMFMRDSK	69
__ 3	Receive the new service.	VMFREC	70
__ 4	Apply the new service.	VMFAPPLY	70
__ 5	Reapply local service by entering local service into the software inventory (if applicable).		71
__ 6	Update Build Status Table.	VMFBLD	73

Table 32. Summary of steps for installing service on High Level Assembler (continued)

Step	Description	VMSES/E Command	Page
__ 7	Rebuild Serviced Parts (objects).	VMFBLD	73
__ 8	Verify the service that has been applied and built.		73
__ 9	Place the service into production.		73

Step 1. Prepare to install service

Carry out preliminary steps prior to receiving service.

Electronic Service (envelope file)

If you have received the service electronically or on CD-ROM, follow the appropriate instructions to retrieve and decompress the envelope files to your A-disk. The decompression is currently done by using the DETERSE MODULE (shipped with VMSES/E).

The documentation envelope and the service envelope files must have a file type of SERVLINK. Make note of the file names that you are using as you will need to enter them in place of the variable *envfilename* in the VMFREC commands that follow.

1. Create a backup copy of the current High Level Assembler before applying the service tape. Save this copy of High Level Assembler until you have completed installing the service and you are confident that the service runs correctly.
2. Log on to the High Level Assembler service user ID: P696234J.
3. Establish read-write (R/W) access to the software inventory disk.

Command	Explanation

`link MAINT 51d 51d mr`
`access 51d d`
	Commands to access the software inventory disk.

4. If you are applying service from tape, attach the tape to the user ID P696234J at virtual address 181. If you have received the service as an Envelope file, ensure the SERVLINK file is available on the A-disk.
5. Establish the correct minidisk access order.

   The VMFSETUP command accesses all the required disks or SFS directories to establish the needed file modes.

Command	Explanation
`vmfsetup P696234J {HLASM|HLASMSFS}`	P696234J is the PPF that is shipped with High Level Assembler. If you have your own PPF override, substitute your PPF name for P696234J. Use `HLASM` if High Level Assembler is installed on minidisk. Use `HLASMSFS` if High Level Assembler is installed in SFS.

6. Receive Documentation. VMFREC with the INFO option loads the documentation and displays a list of all the products on the tape.
   a. If receiving the service from tape:

Command	Explanation
`vmfrec info`	The **info** option loads the documentation (including the product service memo) to the 191 disk and displays a list of products on the tape.

b. If receiving the service from an envelope file:

Command	Explanation
`vmfrec info (env` *envfilename*	The **info** option loads the documentation (including the product service memo) to the 191 disk and displays a list of products. *envfilename* is the file name of the documentation envelope (SERVLINK) file.

7. Check the receive message log ($VMFREC $MSGLOG) for warning and error messages.

Command	Explanation
`vmfview receive`	View the receive message log.

Make a note of which products and components have service on the tape. Press PF5 to show all the status messages that identify the products on the tape.

Step 2. Clear the alternate APPLY disk

Clear the alternate APPLY disk to ensure that you have a clean disk for new service.

1. Clear Disk

   Merge previously applied service to ensure that you have a clean alternate APPLY disk for new service.

Command	Explanation
`vmfmrdsk P696234J {HLASM|HLASMSFS} apply`	Command to clear the alternate APPLY disk. Use `HLASM` if High Level Assembler is installed on minidisk. Use `HLASMSFS` if High Level Assembler is installed in SFS.

2. Check merge message log

Command	Explanation
`vmfview mrd`	Command to review the merge message log ($VMFMRD $MSGLOG). Correct any problems before you go on. For information about specific error messages, refer to *z/VM: Other Components Messages and Codes*.

Step 3. Receive the new service

1. Receive New Service

   Note: If you are installing multiple service tapes, you can receive all the service for this product before applying and building it. For each service tape or electronic envelope you want to receive, do the following:

   a. If receiving the service from tape:

Command	Explanation
vmfrec ppf P696234J {HLASM|HLASMSFS}	Command to receive service from the service tape. All new service is loaded to the alternate DELTA disk. Use HLASM if High Level Assembler is installed on minidisk. Use HLASMSFS if High Level Assembler is installed in SFS.

b. If receiving the service from the PTF envelope file:

Command	Explanation
vmfrec ppf P696234J {HLASM|HLASMSFS} (env *envfilename*	This command receives service from your service envelope. All new service is loaded to the DELTA disk. *envfilename* is the filename of the service (PTF) envelope (SERVLINK) file. Use HLASM if High Level Assembler is installed on minidisk. Use HLASMSFS if High Level Assembler is installed in SFS.

2. Review the receive message log.

Command	Explanation
vmfview receive	Command to review the receive message log ($VMFREC $MSGLOG). Correct any problems before you go on. For information about specific error messages, refer to *z/VM: Other Components Messages and Codes*.

Step 4. Apply the new service

1. Apply the New Service.

Command	Explanation
vmfapply ppf P696234J {HLASM|HLASMSFS}	Command to apply the service you received in “Step 3. Receive the new service” on page 69 The version vector table (VVT) is updated with all service parts and all necessary AUX files are generated on the Alternate Apply disk. Use HLASM if High Level Assembler is installed on minidisk. Use HLASMSFS if High Level Assembler is installed in SFS.

2. Review apply message log

Command	Explanation
vmfview apply	Command to review the apply message log ($VMFAPP $MSGLOG). Correct any problems before going on. For information about specific error messages, refer to *z/VM: Other Components Messages and Codes*.

Step 5. Reapply local service

Do this step only if you received the following message VMFAPP2120W during the VMFAPPLY step.

1. Reapply any local modifications **before** building the serviced High Level Assembler. If a local modification exists for ASMADOPT or one is planned refer to Appendix E, "Local modification procedures (z/VM)," on page 187. For further information refer to *z/VM: Service Guide*.
2. If you are following the process in the *z/VM: Service Guide*, follow the steps that are applicable to your local modification. Then return to this *Installation and Customization Guide* to continue with "Step 6. Update build status table."
3. For the process in *z/VM: Service Guide*, make the following substitutions:
   - `zvm` should be `P696234J`
   - *compname* should be `HLASM` or `HLASMSFS` (minidisk or SFS)
   - *appid* should be `P696234J`
   - *fm-local* should be the file mode of disk 2C2
   - *fm-applyalt* should be the file mode of disk 2A6
   - *outmode localmod* should be `outmode localsam`
   - if necessary, substitute your PPF override in all commands requiring the PPF name.

Step 6. Update build status table

Command	Explanation
`vmfbld ppf P696234J {HLASM|HLASMSFS} (status`	Command to update the Build Status Table. Use `HLASM` if High Level Assembler is installed on minidisk. Use `HLASMSFS` if High Level Assembler is installed in SFS.

Options:

status Identifies build requirements.

If service has been applied to the source product parameter file (file type of $PPF) then carry out the instructions below before going on to "Step 7. Rebuild serviced parts (objects)" on page 73.

If the $PPF files have been serviced the following prompt is displayed:

```
VMFBLD2185R The following source product parameter files have been
serviced:
VMFBLD2185R P696234J $PPF
VMFBLD2185R When source product parameter files are serviced, all
            product parameter files built from them must be recompiled
            using VMFPPF before VMFBLD can be run.
VMFBLD2185R Enter zero (0) to have the serviced source product
            parameter files built to your A-disk and exit VMFBLD so
            you can recompile your product parameter files with VMFPPF.
VMFBLD2185R Enter one (1) to continue only if you have already
            recompiled your product parameter files with VMFPPF.
```

If you select `0` then the following process will recompile the product parameter files (PPF).

You should only select 1 if you have previously compiled your product parameter files. In this case, go to "Step 7. Rebuild serviced parts (objects)" on page 73.

1. Indicate $PPF file needs to be compiled

Command	Explanation
`0`	Enter 0 and continue with the following commands.

The message:

```
VMFBLD2188I Building P696234J $PPF on 191 (A) from level $PFnnnnn
```

is displayed.

2. Compile the product parameter file

Command	Explanation
vmfppf P696234J *	If you have your own PPF override, use your PPF name instead of P696234J.

3. Copy the product parameter file to the software inventory disk.

Command	Explanation
copyfile P696234J $PPF a = = d (olddate replace	**Do not** use your own PPF name in place of P696234J for this COPYFILE command.

Options:

olddate
Uses the date and time on each input file as the date and time of the last update of each corresponding output file.

replace
Causes the output file to replace an existing file with the same file identifier.

4. Erase product parameter file from A disk

Command	Explanation
erase P696234J $PPF a	**Do not** use your own PPF name in place of P696234J for this ERASE command.

5. Update build status table

Command	Explanation
vmfbld ppf P696234J {HLASM|HLASMSFS} (status	VMFBLD updates the build status table. Use `HLASM` if High Level Assembler is installed on minidisk. Use `HLASMSFS` if High Level Assembler is installed in SFS.
1	When you receive the **VMFBLD2185R** prompt, enter 1 to continue.

Options:

status Identifies build requirements.

6. Review the build message log

Command	Explanation
vmfview build	Command to review the build status messages and see which objects need to be built.

Step 7. Rebuild serviced parts (objects)

1. Rebuild serviced parts

Command	Explanation
`vmfbld ppf P696234J {HLASM|HLASMSFS} (serviced`	Command to rebuild serviced parts. If you receive message VMFSBR2000I then you need to rebuild the segment holding High Level Assembler code. **Note:** If your Software Inventory disk (51D) is not owned by the MAINT user ID then make sure the VMSESE PROFILE reflects the correct owning user ID. Use `HLASM` if High Level Assembler is installed on minidisk. Use `HLASMSFS` if High Level Assembler is installed in SFS.

Options:

serviced
Identifies build requirements and builds those objects flagged as **serviced** in the service-level build status table.

2. Review build message log

Command	Explanation
`vmfview build`	Command to review the build message log ($VMFBLD $MSGLOG). Correct any problems before going on. For information about specific error messages, refer to *z/VM: Other Components Messages and Codes*.

Step 8. Verify the service

After you have applied all the files on the service tape, run the installation verification EXEC to ensure that the product functions properly.

Command	Explanation
`V5696234 NOSEG`	Assembles the sample ASMASAMP. The *NOSEG* parameter will cause High Level Assembler to use its code only from disk and not in any segment.

Step 9. Place the service into production

1. If you have a High Level Assembler saved segment in use, you should remove this saved segment before you place the service into production. If you do not have a High Level Assembler saved segment in use, proceed to Step 9, substep 6 on page 75.

Command	Explanation
`q nss all`	Command to display list of system data files that contain named saved systems (NSS) and saved segments.

2. Remove system data file.

Command	Explanation
`purge nss name asmapseg`	Command to remove system data file containing the ASMAPSEG physical saved segment.

3. Re-create the High Level Assembler saved segment.
   a. Prepare to build the segment.
      1) Set storage.

Command	Explanation
def stor 24M	This CP command defines the virtual machine storage to be greater than where the segment is loaded. If defining a saved segment above 16 MB then the virtual machine storage should be above 16 MB. In general the storage should be large enough to hold the segment. If you are using SFS in this user ID, even if only for the A disk, and you are placing the ASMAPSEG segment below 16 MB at 920-9F0 then use a 'DEF STOR 24M' command. This will allow CMS control blocks not to interfere with the storage area to be used by the High Level Assembler saved segment.

      2) IPL the virtual machine **without** accessing the profile.
         If IPLing with minidisks:

Command	Explanation
`ipl cms parm clear nosprof instseg no mtseg no` *** DO NOT press ENTER at the VM READ***	This command clears your virtual machine and bypasses the execution of the system profile (SYSPROF EXEC) and the loading of the installation saved segment (CMSINST).
access (noprof	Bypass the execution of the PROFILE EXEC.

         If IPLing with SFS:

Command	Explanation
`ipl cms parm clear nosprof instseg no mtseg no filepool vmsys` *** DO NOT press ENTER at the VM READ***	This command clears your virtual machine and bypasses the execution of the system profile (SYSPROF EXEC) and the loading of the installation saved segment (CMSINST). This assumes `VMSYS` is the default filepool.
access (noprof	Bypass the execution of the PROFILE EXEC.

      3) Access disks

Command	Explanation
`access 5e5 b`	Access the VMSES/E code.

`link MAINT 51d 51d mr`
 `access 51d d`
	Establish write access to the software inventory disk.

   b. Issue the VMFBLD command to build the High Level Assembler segment.

Command
`vmfbld ppf segbld esasegs segblist ASMAPSEG (serviced`

Options:

serviced
Identifies build requirements and build those objects flagged as **serviced** in the service-level build status table.

4. Review build message log

Command	Explanation
`vmfview build`	Command to review the build message log ($VMFBLD $MSGLOG). Correct any problems before going on. For information about specific build messages, refer to *z/VM: Other Components Messages and Codes*.

5. Update the CMS System disk with the SYSTEM SEGID file.

   The system segment identification file must reside on the CMS system disk (normally file mode S) and must be named SYSTEM SEGID so that it is available to CMS at initialization time. This allows CMS to recognize the logical saved segment name.

   The file must also be copied to the test CMS system disk to prevent backlevelling during application of service.

   The following functions can only be performed by authorized user IDs.

Command	Explanation
`rel d (det`	Release the software inventory disk.

   Log on to MAINT user ID.

   Move SYSTEM SEGID file.

Command	Explanation
`acc 190 t`	Access the production CMS system disk (normally MAINT 190).
`acc 490 v`	Access the test CMS system disk (normally MAINT 490).
`acc 51d d`	Access the software inventory disk.
`copyfile system segid d system segid t2 (olddate replace`	Copy file to production CMS system disk.
`copyfile system segid d system segid v2 (olddate replace`	Copy file to test CMS system disk.

   Options:

   olddate
   Uses the date and time on each input file as the date and time of the last update of each corresponding output file.

   replace
   Causes the output file to replace an existing file with the same file identifier.

6. Move service into production.

   You have now built the segment.

a. Log on to MAINT if you plan to put High Level Assembler general-use code on the 'Y' disk (MAINT's 19E disk). Alternatively, log on to the user ID of the owner of the disk that will contain the production level of the High Level Assembler code.

b. Link to High Level Assembler code and access MAINT's Y disk.

   Choose which access is required depending on whether you have used a minidisk or SFS.

   For minidisk access:

Command	Explanation

`link P696234J 29e 29e rr`
`access 29e e`
	Access High Level Assembler code on a minidisk.

`access 19e f`	Access CMS's Y disk.

   For SFS access:

Command	Explanation
`access vmsys:P696234J.HLASM.TBUILD e`	Access High Level Assembler code on a SFS directory.
`access 19e f`	Access CMS's Y disk.

c. Move High Level Assembler to the Y disk.

   If you have created a saved segment for High Level Assembler, then the only modules that should be copied by the VMFCOPY command are:
   - ASMAHL
   - ASMA90
   - ASMADOPT
   - ASMAINFO

   Therefore the VMFCOPY command must be modified to copy only the four modules which do not reside in the saved segment.

   If you have not created a saved segment then all the High Level Assembler modules should be placed onto the Y disk using the VMFCOPY command.

   In both cases (saved segment or no saved segment), the ASMAMAC MACLIB should be copied.

Command	Explanation
`vmfcopy * MODULE e = = f2 (prodid P696234J%HLASM olddate replace`	Refer to the previous note about which modules should be copied if you have created a logical saved segment.
`vmfcopy * MACLIB e = = f2 (prodid P696234J%HLASM olddate replace`	

   Options:

   olddate
   Uses the date and time on each input file as the date and time of the last update of each corresponding output file.

   replace
   Causes the output file to replace an existing file with the same file identifier.

   If you do not want to use the Y disk for general use code, log on as the owner of the disk where you will put the production level of the High Level Assembler code.

   The VMFCOPY command updates the VMSES PARTCAT file on the 19E disk.

d. Resave the CMS saved system

This updates the shared Y-STAT (the saved Y disk file directory). If you have created a saved segment, it also updates the S-STAT (the saved S disk file directory) to reflect the updated SYSTEM SEGID file.

1) Define named saved system

This creates a named saved system for CMS.

Command	Explanation
`vmfsetup zvm cms`	On z/VM, allows access to the SAMPNSS EXEC which resides on MAINT's 193 disk.
`sampnss cms`	Create named saved system for CMS.

2) Resave the CMS system

Command	Explanation
`ipl 190 clear parm savesys cms`	Resave CMS saved system to reflect the change to the S-STAT (the saved S disk file directory).

Removing High Level Assembler

You use the VMFINS DELETE command to remove High Level Assembler from your system. For information about removing product code and resources from your z/VM system, see the section about the VMFINS DELETE command, in *z/VM: VMSES/E Introduction and Reference*.

Reporting a problem with High Level Assembler

If you have a problem with High Level Assembler, you can first check to see whether the resolution to your problem is already documented, at:

`http://www.ibm.com/software/awdtools/hlasm/support.html`

If you cannot find an answer, report any difficulties with this product to your IBM Support Center. If an APAR is required, the Support Center will provide the address to which any needed documentation can be sent.

To assist with reporting any difficulties refer to the diagnostic process as shown in Chapter 20, "Isolating the problem," on page 115.

Table 33 identifies the component ID (COMP ID) for High Level Assembler for z/VM.

Table 33. Component IDs

COMP ID	Component Name	REL
569623400	VM HIGH LEVEL ASM	360

Obtaining service information

Preventive Service Planning (PSP) information is continually updated as fixes are made available for problems. Check with your IBM Support Center or use IBMLink® (ServiceLink) to see whether there is additional PSP information you need. To obtain this information, specify the following UPGRADE and SUBSET values: `HLASM160` and `HLASMVM360`.

Chapter 11. Planning for installing High Level Assembler on z/VSE

This section contains the following planning information to help you properly install High Level Assembler on z/VSE:
- Worksheet
- What you receive with High Level Assembler
- Choosing required and optional software
- Verifying that you have enough DASD storage
- Deciding where to install
- Checking service updates
- Publications useful during installation

Worksheet: Planning for installing High Level Assembler on z/VSE

Before you begin the installation you should:

1. Determine which of the following you are installing High Level Assembler from:

   A V2 Format tape (with one or more products besides High Level Assembler)

   A tape with just High Level Assembler on it

   See "What you receive with High Level Assembler."
2. Determine the product parts to be installed:

   COMPID *569623400*

   Feature number *5852*

   Tape label *unlabeled*

   See "Basic material" on page 80.
3. Verify that required software (and optional software, if appropriate) is at the level needed. See "What you need to install High Level Assembler" on page 80.
4. Verify that adequate storage is available. See "DASD and other storage required" on page 81.
5. Determine how you are going to install High Level Assembler:

   Using Interactive Interface.

   Using a batch installation job.

   See "Planning where to install High Level Assembler" on page 81.
6. Determine which of the following you want to install in:

   Default library and sublibrary

   A different library and sublibrary

   If using a different library and sublibrary, verify that space is sufficient. See "Planning where to install High Level Assembler" on page 81.
7. Check on latest service updates needed. See "Program support" on page 82.

What you receive with High Level Assembler

You receive the following when you order High Level Assembler for z/VSE:

COMPIDs	Feature Number	System Name
569623400	5852	z/VSE

Distribution media

High Level Assembler is distributed on the following:
- 3480 tape cartridge

The cartridge contains all the programs and data needed for installation.

Basic material

Table 34 describes the program cartridge. Table 35 describes the file content of the program cartridge. z/VSE uses the Maintain System History Program (MSHP) to install this product.

Table 34. Basic material: program cartridge

Medium	Feature Number	Physical Volume	External Label Identification	Volser
3480 cart.	5852	1	HLASM V1R6 for VSE	unlabeled

Table 35. Program cartridge: file content

File	Description
1	Header file containing High Level Assembler copyright statement
2	Backup file ID “HLASM......1.6.0” followed by a MSHP System History File
3	High Level Assembler library file containing the production sublibrary
4	Tape mark
5	End of backup record
6	Tape mark

Optional material

There are no optional machine-readable materials for High Level Assembler.

Cumulative service tape

You might receive an additional tape containing cumulative service with your order. The PTFs on this tape have not yet been incorporated into this release.

Program publications and softcopy

This section identifies the basic and optional publications for High Level Assembler.
- *HLASM Licensed Program Specifications* GC26-4944
- *HLASM Installation and Customization Guide* SC26-3494
- *HLASM Language Reference* SC26-4940
- *HLASM Programmer's Guide* SC26-4941
- *HLASM General Information* GC26-4943

For a list of books for related products, see “Bibliography” on page 191.

Program source materials

There are no source materials available for High Level Assembler.

What you need to install High Level Assembler

The following sections identify the system requirements for installing High Level Assembler.

Required and optional software

This section describes the environment required to install and use High Level Assembler.

High Level Assembler runs on z/VSE with the required licensed program listed in Table 36. **You should install all licensed programs with the minimum release listed or with any subsequent release**.

Table 36. Required programs

Required Licensed Program	Minimum Version Supported
z/VSE	Version 3 Release 1

DASD and other storage required

The DASD storage requirements of High Level Assembler must be added to the storage required by other programs having data in the same library. An estimate of required space is the data set's current allocation plus the storage required by High Level Assembler.

Auxiliary storage is needed for the system history file, for the library and for an intermediate work file (IJSYS03) on a direct-access storage device.

At a minimum, the library must have free the number of library blocks shown in Table 37. A library block equals 1 kilobyte. Therefore, 10 library blocks equal 10,240 bytes.

Table 37 also allows for space for service application.

Table 37. Library requirements

Product Tapefile-id	LIBR BLKS	3380 CYL	3390 CYL	FBA BLKS
HLASM......1.6.0	4000	9	9	8000

If you plan to install this product in a separate MSHP history file, refer to Table 38 for MSHP DASD space requirements.

Table 38. MSHP requirements

File	3380 CYL	3390 CYL	FBA BLKS
IJSYSHF	1	1	900

Planning where to install High Level Assembler

The High Level Assembler default library is PRD1; the default sublibrary is BASE. All High Level Assembler installation jobs assume you are using the sublibrary PRD1.BASE. If you decide to install High Level Assembler in a different library and sublibrary, you need to change some names in the installation jobs.

If you plan to install High Level Assembler to an existing PRD1.BASE sublibrary, make sure there is enough free space to accommodate the additional library blocks.

To check the space, list the directory information of the PRD1 library, using the LISTDIR command of the LIBR program. Make sure there are sufficient library blocks in the free space.

You can use the Interactive Interface to install High Level Assembler, or the documented sample batch job.

Program support

This section describes the IBM support available for High Level Assembler.

Program services

Contact your IBM representative for specific information about available program services.

Preventive Service Planning

Before installing High Level Assembler, you should review the current Preventive Service Planning (PSP) information.

PSP Buckets are identified by UPGRADEs, which specify product levels, and SUBSETs, which specify the CLCs for a product level. The UPGRADE and SUBSET values for High Level Assembler for z/VSE are:

Table 39. PSP upgrade and subset ID

UPGRADE	SUBSET	Description
HLASM160	HLASMVSE689	HLASM VSE

Statement of support procedures

Report any difficulties you have using this program to your IBM Support Center. If an APAR is required, the Support Center will provide the address to which any needed documentation can be sent.

Table 40 identifies the component ID (COMPID) for High Level Assembler for z/VSE.

Table 40. Component IDs

CLC	COMPID	Component Name	RETAIN Release
689	569623400	VSE HIGH LEVEL ASM	689

Program and service level information

This section identifies the program and any relevant service levels of High Level Assembler. The program level refers to the APAR fixes incorporated into the program. The service level refers to the PTFs integrated. Information about the cumulative service tape is also provided.

Program level information

A list of APAR fixes against previous releases of High Level Assembler that have been incorporated into this release is shown in Appendix C, "High Level Assembler Service," on page 183.

Service level information

No PTFs against this release of High Level Assembler have been incorporated into the product tape.

Publications useful during installation

The publications listed in Table 41 may be useful during the installation of High Level Assembler for z/VSE.

Table 41. z/VSE publications

Publication Title	Form Number
z/VSE: Guide to System Functions	SC33-8312
z/VSE: Administration	SC34-2627

Table 41. z/VSE publications (continued)

Publication Title	Form Number
z/VSE: Installation	SC34-2631
z/VSE: Planning	SC34-2635
z/VSE: System Control Statements	SC34-2637
z/VSE: Messages and Codes	SC34-2632 SC34-2633 SC34-2634

Chapter 12. Planning for customizing High Level Assembler on z/VSE

This chapter provides information for planning the customization of High Level Assembler on z/VSE. It includes:
- Deciding whether and what to customize
- Planning to customize the IBM-supplied, default option values
- Planning to install High Level Assembler into the shared virtual area (SVA)

Deciding whether and what to customize

You need to consider whether the IBM-supplied values that come with High Level Assembler suit the needs of your site. These values control such features as:
- Selecting multicultural support
- Assembler options
- Installing High Level Assembler into the SVA

Make sure that High Level Assembler serves the needs of the application programmers at your site. Confer with them while you evaluate the customization options for High Level Assembler, particularly those concerning High Level Assembler options that are also available to the application programmers. Doing so will ensure that the modifications you make best support the application programs being developed at your installation.

The information in this chapter helps you plan your customization. See Chapter 14, "Customizing High Level Assembler on z/VSE," on page 97 for the actual customization procedure.

Selecting multicultural upport

When customizing High Level Assembler, you can choose which language you want to use for diagnostic messages. Languages available are English (mixed case or upper case), German, Spanish, and Japanese. See page "LANGUAGE" on page 154 for information about the LANGUAGE option.

The following combinations are possible:

English Uppercase
Diagnostic messages and listing headings printed in uppercase English.

English Mixed Case
Diagnostic messages and listing headings printed in mixed case English.

German
Diagnostic messages in German and listing headings printed in mixed case English.

Japanese
Diagnostic messages in Kanji and listing headings printed in uppercase English.

Spanish
Diagnostic messages in Spanish and listing headings printed in mixed case English.

Planning to customize High Level Assembler options

The High Level Assembler options should be reviewed to assess the required defaults for your site. A worksheet is provided for planning purposes.

Choices to make now

The following worksheet helps you plan and code the options appropriate for your site. Appendix A, "High Level Assembler Options," on page 143 describes the assembler options, the values that may be specified, and the IBM-supplied default value for each option. Review these default options and complete the worksheet by filling in the **Enter Selection** column.

Table 42. Worksheet options

Object	Enter Selection	IBM-supplied default	Description
ADATA	__________	NO	page 143
ADEXIT	__________	no exit specified	page 144
ALIGN	__________	YES	page 144
ALIGNWARN	__________	YES	page 145
ASA	__________	nothing specified	page 145
BATCH	__________	YES	page 145
CODEPAGE	__________	047C	page 146
COMPAT	__________	NO	page 146
CONTWARN	__________	YES	page 147
DBCS	__________	NO	page 148
DECK	__________	NO	page 148
DELETE	__________	no options deleted	page 149
DSECT	__________	NO	page 150
DXREF	__________	YES	page 150
ESD	__________	YES	page 151
EXLITW	__________	YES	page 151
FLAG	__________	0	page 151
FOLD	__________	NO	page 152
GOFF	__________	NO	page 152
GOFFADATA	__________	NO	page 152
IMPLENWARN	__________	NO	page 153
INEXIT	__________	no exit specified	page 153
INFO	__________	NO	page 154
LANGUAGE	__________	EN	page 154
LIBEXIT	__________	no exit specified	page 155
LIBMAC	__________	NO	page 155
LIMIT	__________	NO	page 156
LINECOUNT	__________	60	page 156
LIST	__________	121	page 157
MACHINE	__________	no default, see OPTABLE	page 158
MACHINELIST	__________	NO	page 159
MAP	__________	YES	page 159
MXREF	__________	SOURCE	page 160
OBJECT	__________	YES	page 160
OBJEXIT	__________	no exit specified	page 161

Table 42. Worksheet options (continued)

Object	Enter Selection	IBM-supplied default	Description
OPTABLE	__________	UNI	page 161
OPTABLELIST	__________	NO	page 163
PAGE0WARN	__________	NO	page 163
PCONTROL	__________	NO	page 164
PESTOP	__________	NO	page 165
PROFILE	__________	NO	page 166
PROFMEM	__________	ASMAPROF	page 166
PRTEXIT	__________	no exit specified	page 167
PUSHWARN	__________	YES	page 167
RA2	__________	NO	page 167
RECORDINFO	__________	YES	page 168
RENT	__________	NO	page 168
RLD	__________	YES	page 169
RXREF	__________	YES	page 169
SECTALGN	__________	8	page 169
SIZE	__________	MAX	page 170
STORAGE	__________	nothing specified	page 171
SUBSTRWARN	__________	NO	page 171
SUPRWARN	__________	NO	page 172
SYSPARMV	__________	none specified	page 172
TERM	__________	NO	page 173
TEST	__________	NO	page 173
THREAD	__________	YES	page 173
TRANSLATE	__________	NO	page 174
TRMEXIT	__________	no exit specified	page 174
TYPECHECK	__________	(MAGNITUDE, REGISTER)	page 175
USING0WARN	__________	YES	page 175
WARN	__________	15	page 176
WORKFILE	__________	NO	page 176
XOBJADATA	__________	NO	page 177
XOBJECT	__________	NO	page 177
XREF	__________	(SHORT,UNREFS)	page 178

Planning to install High Level Assembler into the shared virtual area (SVA)

There are benefits from placing High Level Assembler in the SVA.

Why do it

Installing frequently used data (such as licensed programs) in the SVA has several advantages:

- Several users can access the same physical storage, therefore use of real storage is minimized.
- Using the SVA decreases the input and output rate and the DASD paging requirements, thus improving the performance of the virtual machine.
- The elapsed time for jobs is shorter.

Choices to make now

Table 43 gives the names of the High Level Assembler phases, their approximate sizes, and whether they are eligible for the SVA.

Table 43. High Level Assembler Phases

Member Name	Member Type	SVA Eligible	Approximate Size
ASMA90	PHASE	NO	10K
ASMA93	PHASE	YES	409K
ASMADOPT	PHASE	YES	2K
ASMAMUE	PHASE	YES	23K
ASMALTAS	PHASE	YES	1K
ASMAMDE	PHASE	YES	26K
ASMAMES	PHASE	YES	25K
ASMAMJP	PHASE	YES	25K
ASMAINFO	PHASE	YES	37K
ASMA0474	PHASE	YES	2K
ASMA0475	PHASE	YES	2K
ASMA0476	PHASE	YES	2K
ASMA0477	PHASE	YES	2K
ASMA0478	PHASE	YES	2K
ASMA0479	PHASE	YES	2K
ASMA047A	PHASE	YES	2K
ASMA047B	PHASE	YES	2K
ASMA047C	PHASE	YES	2K
ASMADOP	PHASE	YES	60K
$SVAASMA	PHASE	NO	1K

Phases are placed in the SVA using the SET SDL command. If you have not previously placed High Level Assembler in the SVA, you need to make sure that you have enough free space to accommodate the additional phases.

The maximum number of system directory list (SDL) entries is specified during IPL using the SVA command. Check this number to see if the addition of High Level Assembler phases causes it to be exceeded.

Chapter 13. Installing High Level Assembler on z/VSE

This chapter describes the installation method and the step-by-step procedures you use to install and activate the functions of High Level Assembler.

Overview of installation

You install this release of High Level Assembler by using the Maintain System History Program (MSHP).

Checklist for installing High Level Assembler

Table 44 lists the steps and associated jobs for installing High Level Assembler. The remaining sections in this chapter describe each step. You can use Table 44 as a checklist.

Table 44. Summary of steps for installing High Level Assembler

Step	Description	Installation Job	Page
__ 1	Back up the original system.	—	89
__ 2	Allocate space for the library. (Omit if using the default.)	ASMADEF	89
__ 3	Install High Level Assembler.		91
	Method 1. Install High Level Assembler using the Interactive Interface with High Level Assembler on an Optional Products (V2 Format) tape.		91
	Method 2. Install High Level Assembler using the Interactive Interface with High Level Assembler as the only product on the (V1 Format) tape.		92
	Method 3. Install High Level Assembler using a batch job.	ASMAINST	93
__ 4	Verify the installation of High Level Assembler.	ASMAIVPS	95

Step 1: Back up the original system

Make a backup copy of your current High Level Assembler library or the library you intend to install High Level Assembler into, and the system history file.

For information about backing up libraries and the system history file, see *z/VSE: System Control Statements*.

Step 2: Allocate space for the library (omit if using the default)

By default, High Level Assembler is installed into the PRD1.BASE sublibrary for z/VSE. If you decide to install High Level Assembler into a sublibrary other than PRD1.BASE then proceed with this step.

Decide where to allocate space for the High Level Assembler sublibrary. Identify, on the disk volume (or volumes) to be used for the library, suitable areas of free space. To do this, list the volume table of contents (VTOC) of the disk or disks to be used.

Choose one of the following jobs to list the VTOC:

1. Use the LVTOC utility program. The sample job shown in Figure 18 on page 90 shows the JCL needed to list the VTOC for the volume with serial number SYSWK1.

```
// JOB ASMAVTOC    LIST VOLUME TABLE OF CONTENTS
// ASSGN SYS004,DISK,TEMP,VOL=SYSWK1,SHR
// ASSGN SYS005,SYSLST
// EXEC LVTOC
/*
/&
```

Figure 18. Job to list the contents of a DASD volume

2. Use the DITTO utility program

   As an alternative to using the system utility LVTOC, DITTO's Display VTOC (DVT) may be used. Figure 19 shows a sample job.

```
// JOB ASMADITT    LIST VOLUME TABLE OF CONTENTS
// UPSI 1
// ASSGN SYS001,cuu
// EXEC DITTO,SIZE=512K
$$DITTO DVT INPUT=SYS001,SORTBY=EXTENT
$$DITTO EOJ
/*
/&
```

Figure 19. Job to list the contents of a DASD volume

Use the disk space selected for High Level Assembler in the LIBR installation job to allocate the VSE Librarian library in the sample job shown in Figure 20.

```
// JOB ASMADEF
*  CREATE A LIBRARY FOR THE High Level Assembler
// OPTION LOG
*  Label for the High Level Assembler
*  Library                                          1
// DLBL HLASM,'HLASM.LIBRARY',99/365,SD
// EXTENT SYS002,SYSWK1,,,rtrk,ntrk
// ASSGN SYS002,DISK,VOL=SYSWK1,SHR
* -----------------------------------------
*  Define the High Level Assembler Library          2
* -----------------------------------------
// EXEC LIBR
 DELETE LIB=HLASM
 DEFINE LIB=HLASM
/*
/&
```

Figure 20. Job to allocate the High Level Assembler library space

In area **1** change the *filename* (HLASM in the example) and *file-id* (HLASM.LIBRARY in the example) of High Level Assembler to suit the requirements at your site. Points to consider are:

- The variable *ntrk* indicates the number of tracks required; see Table 37 on page 81 for the number of tracks for your DASD type.
- The variable *rtrk* represents the start position of the extent.

The Librarian job step in area **2** includes a DELETE statement before the DEFINE statement so the job can be rerun. This means the following messages are issued when the job runs for the first time; however these may be treated as informational and ignored. The job continues to allocate the library.

The messages are:

```
L101I  LIBRARY HLASM DOES NOT EXIST
L027I  ABNORMAL END DURING DELETE COMMAND
       PROCESSING
L113I  RETURN CODE OF DELETE IS 8
```

Step 3: Install High Level Assembler

You can install High Level Assembler using either the Interactive Interface of z/VSE or a batch installation job.

Method 1: Install High Level Assembler using the interactive interface with V2 Format tape

Carry out the following tasks:

1. Mount the High Level Assembler tape on an available tape drive.
2. Library Labels

   If the product is not being installed in the default library then the library to contain High Level Assembler must have its DLBL and EXTENT in the label information area.
3. Log on to the Interactive Interface

   To install High Level Assembler using the Interactive Interface, log on to the z/VSE Interactive Interface as the system administrator. For more information about the functions of the Interactive Interface, refer to *z/VSE: Guide to System Functions*.

   In the following menus, enter the highlighted items that appear after the ===> symbol.
4. Mount the High Level Assembler tape on an available tape drive.
5. In the **z/VSE FUNCTION SELECTION** menu, select: ===> ***1*** (Installation)
6. In the **INSTALLATION** menu:

   You received High Level Assembler as a V2 Format tape, which contains one or more optional products, therefore select: ===> ***1*** (Install Programs - V2 Format)
7. In the **INSTALL PROGRAMS - V2 FORMAT** menu, select: ===> ***1*** (Prepare for Installation)
8. In the **PREPARE FOR INSTALLATION** menu, select: ===> ***cuu*** (the address of the tape drive where you mounted the distribution tape.)

   (If you are using a virtual tape, enter ***1*** for Virtual Tape, otherwise enter **2**.)
9. In the **JOB DISPOSITION** menu:

   Make any changes required and press Enter to submit the job.
10. Respond to console messages:

    When the job starts it asks if the tape is ready. After the tape is scanned it asks if further tapes are to be read. Refer to Figure 21 for examples of the console messages.

```
01 BG 000 IESI0091I PLEASE MOUNT TAPE LABELLED "VSE OPTIONAL TAPE NUMBER
02*BG 000 IESI0092A MOUNT ON TAPE DRIVE 580 . WHEN READY, REPLY "END/ENTER"
03*BG-000
04 0
05*BG 000 IESI0090A ARE THERE ANY MORE OPTIONAL PROGRAM TAPES? YES/NO
06*BG-000
10 0 no
11 BG 000 EOJ INSPRE    MAX.RETURN CODE=0000
```

Figure 21. Console messages

The output listing from this job gives a list of the optional programs on the distribution tape with program identifiers and recommended library sizes. The tapefile identifier for High Level Assembler is `HLASM......1.6.0`.

The program identifiers of the optional programs on the distribution tape are also automatically entered on the **INSTALL ADDITIONAL PROGRAM(S) FROM TAPE** menu.

11. To return to the **z/VSE FUNCTION SELECTION** menu, enter: ===> ***1*** (Installation)
12. In the **INSTALLATION** menu, select: ===> ***1*** (Install Programs - V2 Format)
13. In the **INSTALL PROGRAMS - V2 FORMAT** menu, select: ===> **2** (Install Programs(s) from Tape)
14. In the **INSTALL ADDITIONAL PROGRAM(S) FROM TAPE** menu:

    Enter **1** (install) in the OPT field against the tapefile identifier ***HLASM......1.6.0*** (High Level Assembler) and **2** (skip installation) against any other optional products you do not intend to install at this time.

    If you did not use the default library PRD1.BASE, enter the name of your library and sublibrary on this screen. The DLBL and EXTENT information for this library should already be in the label information area.

    Press PF5 to generate the installation job.
15. Retain products list

    Decide if you want to keep the product list previously generated from LIBR utility scan of the product tape.
16. In the **INSTALL ADDITIONAL PROGRAM(S) FROM TAPE** menu, enter: ===> ***cuu*** (the address of the tape drive where you mounted the High Level Assembler tape)

    (If you are using a virtual tape, enter ***1*** for Virtual Tape, otherwise enter **2**.)
17. In the **JOB DISPOSITION** menu:

    Make any changes required and press ENTER to submit the job to install High Level Assembler.
18. Respond to console messages

    Confirm prompt when asking for tape to install product.

Condition code and messages

If you do not receive a condition code of 0:

1. Check the list output for error conditions.
2. See the *z/VSE: Messages and Codes* for corrective action.
3. Correct the error.
4. Rerun the job.
5. Recheck the condition code.

Method 2: Install High Level Assembler using the interactive interface with V1 Format tape

Carry out the following tasks:

1. Log on to the Interactive Interface

   To install High Level Assembler using the Interactive Interface, log on to the z/VSE Interactive Interface as the system administrator. For more information about the functions of the Interactive Interface, refer to *z/VSE: Guide to System Functions*.

   In the following menus enter the highlighted items that appear after the ===> symbol.
2. Mount the High Level Assembler tape on an available tape drive.
3. In the **z/VSE FUNCTION SELECTION** menu, select: ===> ***1*** (Installation)
4. In the **INSTALLATION** menu:

   You received High Level Assembler as a V2 Format tape, which contains one or more optional products, select: ===> **2** (Install Programs - V1 Format)

5. In the **INSTALL PROGRAMS - V1 FORMAT** menu select: ===> ***HLASM......1.6.0*** (Tapefile-id)

   If you did not use the default library PRD1.BASE, enter the name of your library and sublibrary on this screen. The DLBL and EXTENT information for this library should already be in the label information area.

   Press PF5 to generate the installation job.
6. In the **INSTALL ADDITIONAL PROGRAM(S) FROM TAPE** menu, enter: ===> ***cuu*** (the address of the tape drive where you mounted the High Level Assembler tape)

   (If you are using a virtual tape, enter ***1*** for Virtual Tape, otherwise enter **2**.)
7. In the **JOB DISPOSITION** menu:

   Make any changes required and press Enter to submit the job to install High Level Assembler.
8. Respond to console messages

   Confirm prompt when asking for tape to install product.

Condition code and messages

If you do not receive a condition code of 0:

1. Check the list output for error conditions.
2. See the *z/VSE: Messages and Codes* for corrective action.
3. Correct the error.
4. Rerun the job.
5. Recheck the condition code.

Method 3: Install High Level Assembler using a batch job

The batch installation job stream for installing High Level Assembler uses the `MSHP` system history file that already exists as part of the z/VSE system. This system history file might already be defined in the system standard labels; if not, make sure that `DLBL` and `EXTENT` statements, with the necessary information for the system history file, are included in the job stream.

Depending on how you request the High Level Assembler product you might receive different installation tapes. One could contain only the High Level Assembler product, the other might be a V2 Format tape containing one or more optional program products. The job shown in Figure 22 on page 94 handles both types of tape (V2 Format and V1 Format).

Create and tailor the following job stream, mount the distribution tape, and run the installation job.

Figure 22 on page 94 provides the JCL required to install High Level Assembler. Tailor this JCL to suit the requirements at your site.

As many as five modifications might be required to tailor the JCL. The keys in Figure 22 on page 94 are explained individually and refer to the sections that accompany the JCL description.

```
// JOB ASMAINST
*  INSTALL THE High Level Assembler LIBRARY
// OPTION LOG
*  Label for High Level Assembler Library       1
*  Assign  install tape as SYS006               2
// ASSGN SYS006,cuu
// MTC REW,SYS006
* ---------------------------------------
*  This step installs High Level Assembler
*  from the distribution tape
*  using the VSE system history file            3
* ---------------------------------------
// EXEC MSHP,SIZE=900K,PARM='PIDSTACKED'
 INSTALL PROD FROMTAPE ID='HLASM......1.6.0' -
    PROD INTO=PRD1.BASE
/*
* ---------------------------------------
*  List the High Level Assembler Library        4
* -------------------------------------
// EXEC LIBR
 LISTDIR SUBLIB=PRD1.BASE  -
     OUTPUT=NORMAL -
     UNIT=SYSLST
/*
* -----------------------------------------
*  Retrace the High Level Assembler product     5
* -----------------------------------------
// EXEC MSHP,SIZE=900K
RETRACE COMPONENT IDENTIFIER=5696-234-00
/*
// MTC RUN,SYS006
/*
/&
```

Figure 22. Job to install High Level Assembler

1. Specify the Label Information

   In area **1**, if you are installing High Level Assembler into a sublibrary other than the default then insert `DLBL`, `EXTENT`, and `ASSGN` information as specified in Figure 20 on page 90. The library name must match the name used in the allocation job in Figure 20 on page 90.

   There is no `DLBL` statement for the system history file. Typically it has a permanent system standard label for this, with `IJSYSHF` as the file name. (`IJSYSHF` is the default file name that `MSHP` looks for in a label statement.)

2. Assign the Distribution Tape

   Assign the distribution tape in area **2** to logical unit `SYS006`. Replace *cuu* with the address of the tape drive on which the distribution tape is mounted. Alternatively you may use the generic tape assignment:

   ```
   // ASSGN SYS006,TAPE
   ```

3. Install High Level Assembler

   Area **3** of the job calls `MSHP` to install High Level Assembler into the library and sublibrary identified on the `INTO` operand of the `INSTALL` statement. If you are installing High Level Assembler into a library and sublibrary other than the default, then change the name of the library and sublibrary on the `INTO` operand of the INSTALL statement to reflect the values you specified in area **3**. For more information about the install options, refer to *z/VSE: System Control Statements*.

4. List the Directory Entries

The step in area **4** of the job lists the directory entries of the sublibrary where High Level Assembler was installed. Remove this step if a directory list is not required. If you have installed High Level Assembler into a sublibrary other than the default, then the name of the sublibrary must be changed to reflect the value you specified in area **3**.

Entries for High Level Assembler have a four character prefix of ASMA to distinguish them from other products; there are three exceptions to this rule:

- `HD234689.Z`
- `$SVAASMA.PHASE`
- `$SVAASMA.OBJ`

5. Retrace the High Level Assembler product in the system history file.

   The final step in area **5** of the job prints the component records from the system history file for High Level Assembler. Remove this step if a retrace listing is not required.

   If this job has to be run again, remember first to restore the system history file, which should have been backed up before running this install job, and second to run the library allocation step again, if applicable.

Step 4: Verify the installation of High Level Assembler

Two sample High Level Assembler source decks, and a sample job are provided on the distribution tape, to help you verify your installation. These sample source decks are `ASMASAM1.A` and `ASMASAM2.A`. The sample job is `ASMAIVPS.Z`.

To verify your installation, run the sample job ASMAIVPS.Z. This job will verify your installation by exercising representative features of High Level Assembler.

Figure 23 shows the job `ASMAIVPS.Z` provided in the installed sublibrary. It runs the sample program from the same sublibrary. If you installed High Level Assembler in a sublibrary other than the default, modify the IBM-supplied `JCL` that runs the verification program.

```
// JOB ASMAIVPS
*
*  SAMPLE JCL TO VERIFY INSTALLATION OF THE
*  High Level Assembler.
*
// LIBDEF *,SEARCH=(lib.sublib,PRD1.BASE)    1
// EXEC ASMA90,SIZE=ASMA90
     COPY ASMASAM1
     COPY ASMASAM2
/*
/&
```

Figure 23. Job to verify the success of your installation

In area **1**, specify the user library where the source format macros have been placed, followed by the library where the High Level Assembler resides.

If you are using the library exit for processing E-decks then modify the EXEC statement in Figure 23 to:

```
// EXEC ASMA90,SIZE=(ASMA90,128K),PARM='EXIT(LIBEXIT(EDECKXIT))'
```

A return code of 0 (zero) for the job indicates that the sample program completed successfully. The assembly of the sample program is the actual verification that the product is installed and functions correctly.

The program being assembled is not intended to be run; however, if it is run, it sets a return code of zero and returns to the caller.

Chapter 14. Customizing High Level Assembler on z/VSE

You can customize, or modify, High Level Assembler only after installing the product. This chapter includes:
- Changing default options
- Placing High Level Assembler into the Shared Virtual Area

Changing default options

If you decided to change any of the default High Level Assembler options when planning to customize High Level Assembler (see Chapter 12, "Planning for customizing High Level Assembler on z/VSE," on page 85) you can change these options now.

A sample job `ASMAOPTV.Z` is provided to help you modify and assemble the IBM-supplied, default assembler options. It is shown in Figure 24 on page 98.

The ASMAOPT macro in `ASMAOPTV.Z` is used to specify the options you want to set as the defaults. Refer to "General rules for coding the ASMADOPT ASSEMBLE file" on page 63 for more information about how to enter the options into the sample job. For more detailed information about these options see the *HLASM Programmer's Guide*. For a description of the options and the values you can specify, see the Appendix A, "High Level Assembler Options," on page 143.

When the changes have been made, submit the job for assembly on your z/VSE system. The ASMAOPTV job assembles and catalogs a new ASMADOPT.OBJ module in the user sublibrary. When this job has completed successfully, continue with the section on re-linking High Level Assembler phase `ASMADOPT`, which completes the customization of the High Level Assembler options.

```
// JOB ASMAOPTV ASSEMBLE DEFAULT OPTIONS
*
// SETPARM VOLUME='SYSWK2'              * VOLUME TO USE          |
// SETPARM START='????'                 * STARTING EXTENT TRK/BLK | 1
// SETPARM LENGTH='0020'                * LENGTH OF WORK FILE     |
*
*  Assemble High Level Assembler Default Options
*
// LIBDEF *,SEARCH=(PRD1.BASE)                                       2
// DLBL IJSYSPH,'ASSEMBLE.OUTPUT',0
// EXTENT SYSPCH,&VOLUME,1,0,&START,&LENGTH
ASSGN SYSPCH,DISK,VOL=&VOLUME,SHR
// OPTION DECK
// EXEC ASMA90
         PUNCH 'CATALOG ASMADOPT.OBJ REPLACE=YES'
         PRINT ON,GEN
         ASMAOPT                                                     3
         END
/*
CLOSE SYSPCH,PUNCH
// DLBL IJSYSIN,'ASSEMBLE.OUTPUT',0
// EXTENT SYSIPT
ASSGN SYSIPT,DISK,VOL=&VOLUME,SHR
// EXEC LIBR,PARM='MSHP;ACCESS SUBLIB=lib.sublib'                    4
/*
CLOSE SYSIPT,SYSRDR
/*
/&
// JOB RESET
ASSGN SYSIPT,SYSRDR    IF 1A93D, CLOSE SYSIPT,SYSRDR
ASSGN SYSPCH,PUNCH     IF 1A93D, CLOSE SYSPCH,PUNCH
/*
/&
```

Figure 24. Sample JCL to assemble the default options

In area **1**, change the SETPARM values to those applicable to your installation.

In area **2**, if High Level Assembler has been installed in a different sublibrary from the default, change this to reflect your sublibrary.

In area **3** include only the options and values you want to change.

In area **4**, change the sublibrary to reflect a user sublibrary in which changes to the High Level Assembler default options will be cataloged. This should be in a user library to ensure that the original IBM-supplied defaults are not overwritten.

Relink High Level Assembler phase ASMADOPT

The High Level Assembler phase ASMADOPT needs to be relinked to successfully change the default options. The sample JCL member ASMAOPTL.Z is provided, which relinks the High Level Assembler phase ASMADOPT (refer to Figure 25 on page 99).

```
// JOB ASMAOPTL   Linkedit Default Options Phase
*
*  link edit ASMADOPT
*
// LIBDEF *,SEARCH=(lib.sublib,PRD2.CONFIG,PRD1.BASE)  1
// LIBDEF PHASE,CATALOG=lib.sublib                      2
// OPTION CATAL
 INCLUDE ASMADOLK                                       3
// EXEC LNKEDT,PARM='MSHP'
/*
/&
```

Figure 25. Sample JCL to linkedit the default options phase

In area **1** the library-search chain points first to the sublibrary where ASMADOPT.OBJ was catalogued when you assembled the changed options. It then points to the sublibrary where High Level Assembler was installed, and where the link book ASMADOLK.OBJ resides.

In area **2**, change the library and sublibrary to reflect the user sublibrary where ASMADOPT.PHASE will reside.

In area **3** a link book supplies the phase statement and include values.

Placing High Level Assembler into the Shared Virtual Area (SVA)

This customization must be done by a system administrator.

Table 43 on page 88 gives a list of phase names, their link-edit attributes and approximate sizes. It shows which phases are eligible for sharing storage, and the approximate size of each phase.

To assist you in placing the High Level Assembler phases into the SVA, a sample member named `ASMASVA2.Z` is provided. This sample job includes all the High Level Assembler phases eligible for the SVA.

z/VSE: System Control Statements provides specific instructions about placing phases into the SVA.

The phases are loaded using a load list. The $SVAASMA load list is provided. This load list can also be placed in the BG startup procedure. If this is done then the library containing the High Level Assembler must be in the LIBDEF search chain within the LIBSDL procedure.

```
// JOB ASMASVA2 LOAD SVA-ELIGIBLE PHASES
*
*  Load SVA using a load list
*
// LIBDEF *,SEARCH=(lib.sublib,PRD1.BASE)
SET SDL
LIST=$SVAASMA
/*
/&
```

Figure 26. Sample JCL to load SVA-eligible phases using load list

On the LIBDEF statement, change `lib.sublib` to your sublibrary if you have installed any phases into your own sublibrary. For example, you may have installed your own version of ASMADOPT into your own sublibrary (see "Changing default options" on page 97).

Chapter 15. Maintaining High Level Assembler on z/VSE

This chapter describes how to re-install or remove High Level Assembler and how to apply service updates. To use the maintenance procedures, you must have already installed High Level Assembler and any required products.

Re-installing High Level Assembler

You do not need to perform all the planning and installation procedures to re-install High Level Assembler. For example, you might not need to reconsider your storage needs if High Level Assembler replaces the existing High Level Assembler data sets.

You do not need to remove High Level Assembler from your system before re-installing High Level Assembler, unless you intend to re-install the product in a different sublibrary from the previous installation. In this case you must remove High Level Assembler from the system history file before you can re-install it.

To re-install High Level Assembler, you follow the same steps as for installing High Level Assembler. See Chapter 13, "Installing High Level Assembler on z/VSE," on page 89.

Applying service updates

You might need to apply maintenance or service updates to High Level Assembler periodically. This section details these procedures.

What you receive

If you report a problem with High Level Assembler to your IBM Support Center, you will receive a tape containing one or more APARs or PTFs to solve your problem.

You might also receive a list of pre-requisite APARs or PTFs, which should have been applied to your system before applying the current service. These pre-requisite APARs or PTFs might relate to High Level Assembler or any other licensed product you have installed, including z/VSE.

You apply service to High Level Assembler using either the z/VSE Interactive Interface or a batch job.

The following checklist provides a summary of steps you should use to apply service to High Level Assembler.

Checklist for applying service

Table 45 lists the steps for installing corrective service on High Level Assembler. You can use Table 45 as a checklist.

Table 45. Summary of steps for installing service on High Level Assembler

Step	Description	MSHP Command or Job name	Page
__ 1	Ensure prerequisite APARs or PTFs are applied.	RETRACE	102
__ 2	Backup existing system.		102
__ 3	Apply service.	INSTALL	102
__ 4	Run the installation verification program.	ASMAIVPS	103

Step 1. Check prerequisite APARs or PTFs

Prerequisite APARs or PTFs need to be applied to your system before you can apply the current maintenance. These APARs or PTFs might apply to High Level Assembler or any licensed program you have installed at your site.

Your IBM Support Center has given you a list of any relevant prerequisite APARs or PTFs. Probably most have been applied to your system. You can verify this by retracing the APARs and PTFs in your system history file. The job shown in Figure 27 shows how to retrace APARs and PTFs in the system history file. This job is supplied as ASMARETR.Z

Use this listing to check that you have already applied any pre-requisite APARs or PTFs. If you have not, arrange for your IBM Support Center to send them to you and apply them before applying other service.

```
// JOB ASMARETR Retrace APARs and PTFs
// EXEC MSHP,SIZE=700K
RETRACE APARS
RETRACE PTFS
/*
/&
```

Figure 27. Job to retrace APARs and PTFs

Step 2. Backup original system

Make a backup copy of your current High Level Assembler library and the system history file. For information about backing up libraries and the system history file, see *z/VSE: System Control Statements*.

Step 3. Apply service

You can apply service to High Level Assembler from the provided service tape using either the Interactive Interface or a batch job.

Method 1: Apply service using the interactive interface

To apply service to High Level Assembler using the Interactive Interface, log on to the z/VSE Interactive Interface as the system administrator. (For more information about the functions of the Interactive Interface, refer to *z/VSE: Administration*.)

Mount the service tape on an available tape drive.

In the following menus specify the highlighted items that appear after the ===> symbol.

1. **z/VSE FUNCTION SELECTION** menu: ===> ***1*** (Installation)
2. **INSTALLATION** menu: ===> ***4*** (IBM Service)
3. **IBM SERVICE** menu: ===> **2** (PTF Handling)
4. **PTF HANDLING** menu:
   - If you want to print the documentation about the supplied PTFs before applying the service, select: ===> ***1*** (Print Service Document)

     PRINT SERVICE DOCUMENT menu: ===> ***1*** (Service medium (tape)) ===> ***cuu*** (the address of the tape drive where you mounted the service tape.)

     Enter ***1*** or ***2*** to select the type of document you want to print.
   - If you want to apply the service directly, select: ===> ***3*** (Apply PTFs)

     APPLY PTF menu: ===> ***1*** (Service medium (tape)) ===> ***cuu*** (the address of the tape drive where you mounted the service tape). Press Enter. ===> ***SERVICE UNIT NAME*** (allocate a unique name for the service unit) Press Enter.

5. **JOB DISPOSITION** menu:

   Make any changes required and press Enter to submit the job and apply the service.
6. Respond to console messages

   Confirm prompt when asking for service tape.

Method 2: Apply service using a batch job

The batch job to apply service to High Level Assembler uses the MSHP system history file where High Level Assembler was installed.

A sample job to apply service using MSHP is shown in Figure 28. This job is supplied as ASMAAPP.Z. For more information about MSHP see *z/VSE: System Control Statements*.

```
// JOB ASMAAPP Apply Service
// ASSGN SYS006,cuu              1
// EXEC MSHP,SIZE=700K
INSTALL SERVICE FROMTAPE         2
/*
/&
```

Figure 28. Job to install service

In area **1**, change *cuu* to the address of the tape drive where you have mounted the service tape.

Area **2** shows the MSHP statement to install service from a tape. The information in the system history file directs MSHP to apply the service to the sublibrary in which High Level Assembler is installed. You do not need to supply this information.

Step 4. Run the installation verification program (IVP)

After you have applied all the files on the service tape, run the installation verification program, ASMAIVPS, to ensure that High Level Assembler functions properly. Refer to "Step 4: Verify the installation of High Level Assembler" on page 95.

Removing High Level Assembler

You do not have to remove High Level Assembler from your system before installing a new version or release.

If you do have to remove High Level Assembler for any reason, you must delete all the High Level Assembler entries from your sublibrary and remove High Level Assembler from the system history file.

A sample job, ASMADELV.Z, is supplied to help you do this. This job will:
- delete all High Level Assembler members from the sublibrary where it is installed
- remove High Level Assembler from the System History File
- update DTRIHIST.Z in the system residence library to remove High Level Assembler from the list of installed products

  Consult the instructions in the job for more information.

To report a problem with High Level Assembler

If you have a problem with High Level Assembler, you can first check to see whether the resolution to your problem is already documented, at:

`http://www.ibm.com/software/awdtools/hlasm/support.html`

If you cannot find an answer, report any difficulties with this product to your IBM Support Center. If an APAR is required, the Support Center will provide the address to which any needed documentation can be sent.

To assist with reporting any difficulties refer to the diagnostic process as shown in Chapter 20, “Isolating the problem,” on page 115.

Table 46 identifies the component ID (COMP ID) for High Level Assembler for z/VSE.

Table 46. Component IDs

COMP ID	**Component Name**	**REL**
569623400	VSE HIGH LEVEL ASM	689

Obtaining service information

Preventive Service Planning (PSP) information is continually updated as fixes are made available for problems. Check with your IBM Support Center or use either Information/Access or SoftwareXcel Extended to see whether there is additional PSP information that you need. To obtain this information, specify the following UPGRADE and SUBSET values: HLASM160 and HLASMVSE689.

Chapter 16. Planning for installing High Level Assembler on zLinux

This section contains the following planning information to help you properly install High Level Assembler on zLinux:
- An introduction to High Level Assembler on the zLinux platform
- What you receive with High Level Assembler
- Required and optional software
- Machine requirements
- Storage requirements

Introduction to High Level Assembler on zLinux

The IBM High Level Assembler for Linux on zSeries®, is an IBM licensed program that can be used to assemble assembler language programs that use the following machine instructions:
- System/370
- System/370 Extended Architecture (370-XA)
- Enterprise Systems Architecture/370 (ESA/370)
- Enterprise Systems Architecture/390 (ESA/390)
- z/Architecture® machine instructions

High Level Assembler on zLinux is installed using the Redhat Package Manager (RPM).

What you receive with High Level Assembler

This section describes the "Distribution medium" and "RPM package contents."

Distribution medium

High Level Assembler is distributed on a 3480 tape cartridge.

The cartridge contains all the programs and data needed for installation. It is labeled and has a VOLSER of 5799TC. There are four files on the 3480 tape; they are described in Table 47.

Table 47. 3480 cartridge content

No.	DSNAME	RECFM	LRECL	BLKSIZE	Description
1	README	FB	80	27920	This file
2	H3100300.PDF	U	0	32760	IPLA Licence for 5799-TCQ
3	ASML1020.PDF	U	0	32760	User's Guide
4	ASMA90.RPM	U	0	32760	RPM Package

RPM package contents

The product files contained within ASMA90.RPM itself are:

Filename	**Description**
asma90	The HLASM executable file
asma90.map	Map file for the HLASM executable

Filename	Description
asmalib	Additional and alternate module library: • OBJ-to-ELF Converter exit • APAR data file used by the INFO assembler option • Alternate language message files • Alternate codepage files
asmalib.map	Map file for additional and alternate module library
asml1020.pdf	User's Guide
h3100300.pdf	License Information
README	Information about the product
runasma90	Sample execution script

What you need to install High Level Assembler

The following sections identify the system requirements for installing High Level Assembler.

Required and optional software

High Level Assembler runs under the operating systems listed below. Unless otherwise stated, the assembler also operates under subsequent versions, releases, and modification levels of these systems:
- Linux on zSeries (with 32-bit compatibility environment)
- Linux on S/390

High Level Assembler supports the operation codes available with the following mode processors:
- System/370 Architecture
- Extended Architecture (370-XA)
- Enterprise Systems Architecture/370 (ESA/370)
- Enterprise Systems Architecture/390 (ESA/390)
- z/Architecture

Machine requirements

For assembling High Level Assembler programs

Programs written using High Level Assembler can be assembled, including use of the z/Architecture processor machine instructions, the Extended Architecture mode processor machine instructions, and Enterprise System Architecture mode processor machine instructions, on all System/370 family and its follow-on machines supporting the following operating systems:
- Linux for zSeries
- Linux for S/390

You might require an operating system-specific macro library to assemble programs that run under that operating system, depending on macro usage.

For running High Level Assembler programs

A generated object program using z/Architecture, Extended Architecture (370-XA), Enterprise Systems Architecture/370 (ESA/370), Enterprise Systems Architecture/390 (ESA/390), Enterprise Systems/9000 (ES/9000) or Vector instructions can be run only on an applicable processor under an operating system that provides the necessary architecture support for the instructions used.

Tape device

High Level Assembler is distributed on 3480 tape cartridge.

An appropriate tape device is required for installation.

Storage requirements

Virtual storage
High Level Assembler requires a minimum of 650K bytes of main storage. 450K bytes of storage are required for High Level Assembler modules. The rest of the storage allocated to the assembler is used for assembler working storage.

Auxiliary storage space
Depending on the assembler options used, auxiliary storage space might be required for the following data sets:
- System input
- Macro instruction library—either system or private or both
- Print output
- Object module output

Library space
The package files will require 742K bytes of disk storage for modules and procedures.

Chapter 17. Installing High Level Assembler on zLinux

This chapter describes the installation method and the step-by-step procedures you use to install High Level Assembler on zLinux.

You install High Level Assembler on zLinux using the Redhat Package Manager (RPM). You first transfer the product files from the 3480 tape cartridge to a host system (z/OS, z/VM, or z/VSE), and then across to the Linux system which will be the target of the RPM installation.

The file transfer process

There are four files on the 3480 tape which should be uploaded to the host system. The four files are sequential data sets which can be copied to DASD using a utility such as IEBGENER (for z/OS), or equivalent utilities for z/VM or z/VSE. See "Sample JCL" for sample job control to copy the files to a z/OS system.

Once the files are on DASD, then normal ftp can be used to transfer them to the zLinux system. The type of transfer and resulting Linux file names are:

Host DSNAME	Transfer Type	Linux Filename
README	ASCII	README
H3100300.PDF	Binary	h3100300.pdf
ASML1020.PDF	Binary	asml1020.pdf
ASMA90.RPM	Binary	asma90-1.6.0-3.s390.rpm

The RPM package ASMA90.RPM must be renamed during or after the file transfer so that RPM will correctly recognize the package.

The RPM package is the only file that is actually needed to be present on the zLinux system for the RPM installation to succeed. The other files can be transferred to any suitable computer platform which allows viewing of text and pdf files.

Sample JCL

The following JCL can be used to copy the files from the 3480 tape cartridge up to a host z/OS system. Add an appropriate JOB statement for the host system and change 'hlq' to an appropriate data set qualifier.

```
//*
//FILE01   EXEC PGM=IEBGENER
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SYSUT1   DD DSN=README,DISP=SHR,
//            UNIT=3480,VOL=(PRIVATE,RETAIN,,,SER=5799TC),
//            LABEL=(1,SL),EXPDT=98000
//*
//SYSUT2   DD DSN=hlq.README,DISP=(NEW,CATLG),
//            SPACE=(1,(100,100),RLSE),AVGREC=K
//*
//*
//FILE02   EXEC PGM=IEBGENER,COND=(4,LT)
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

```
//*
//SYSUT1   DD DSN=H3100300.PDF,DISP=SHR,
//            UNIT=3480,VOL=(PRIVATE,RETAIN,REF=*.FILE01.SYSUT1),
//            LABEL=(2,SL),EXPDT=98000
//*
//SYSUT2   DD DSN=hlq.H3100300.PDF,DISP=(NEW,CATLG),
//            SPACE=(1,(1000,1000),RLSE),AVGREC=K
//*
//*
//FILE03   EXEC PGM=IEBGENER,COND=(4,LT)
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SYSUT1   DD DSN=ASML1020.PDF,DISP=SHR,
//            UNIT=3480,VOL=(PRIVATE,RETAIN,REF=*.FILE01.SYSUT1),
//            LABEL=(3,SL),EXPDT=98000
//*
//SYSUT2   DD DSN=hlq.ASML1020.PDF,DISP=(NEW,CATLG),
//            SPACE=(1,(1000,1000),RLSE),AVGREC=K
//*
//*
//FILE04   EXEC PGM=IEBGENER,COND=(4,LT)
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SYSUT1   DD DSN=ASMA90.RPM,DISP=SHR,
//            UNIT=3480,VOL=(PRIVATE,RETAIN,REF=*.FILE01.SYSUT1),
//            LABEL=(4,SL),EXPDT=98000
//*
//SYSUT2   DD DSN=hlq.ASMA90.RPM,DISP=(NEW,CATLG),
//            SPACE=(1,(1000,1000),RLSE),AVGREC=K
//*
```

Installation

The install should be a standard rpm install, for example:

```
rpm -v --prefix /usr/ --install asma90-1.6.0-3.s390.rpm
```

The rpm package is relocatable, so the --prefix option can be used to install High Level Assembler into a different path than the default /usr path if required.

Prior to the install you can query the rpm package. Here are some examples.

- To list the spec file:

  ```
  rpm -v --query --info --package asma90-1.6.0-3.s390.rpm
  ```

- To list the files within the rpm package:

  ```
  rpm -v --query --list --package asma90-1.6.0-3.s390.rpm
  ```

If there are any install problems then the files can be manually extracted from the rpm package, for example:

```
mkdir cpio
cd cpio
rpm2cpio asma90-1.6.0-3.s390.rpm | cpio --extract --verbose --make-directories --preserve-modification-time
```

Chapter 18. Starting High Level Assembler on zLinux

This chapter describes how you start the assembler on zLinux.

Starting the assembler

To start the assembler enter the following (the line spacing is only for readability; enter the command on a single line):

```
./asma90   input_path_name
       -l   list_path_name
       -o object_path_name
       -t   term_path_name
       -L syslib_path_names
       -E object_exit_path_name
```

1. `input_path_name` is not prefixed by an identifier while the other dataset names are.
2. The identifiers are:
   `-l` for the listing output
   `-o` for the GOFF/object output
   `-t` for the TERM output
   `-L` for the SYSLIB datasets
   `-E` for exits when the elf32 option is supplied (see below)

   Identifiers are case sensitive.
3. Options are specified as `--options='options'`.

   The options identifier must be prefixed by two hyphens and the options enclosed in apostrophes.
4. It is not necessary to enter the full path name for any file. If the current directory is to be used, no directory information is required except for SYSLIB, where `./` must be supplied.
5. Multiple SYSLIB path names can be supplied. Each path name is separated by a colon (for example, `path_name_1:path_name_2:...`).
6. SYSLIB path name extensions are supported. The extension is specified as `path_name/*.ext`. If more than one extension is required, this can be specified as `path_name/*.ext1:/path_name/*ext2:...`.

   You may need to quote the library directory name, as in `'./*.mac'`, or specify a set of specifications separated by colons, as in `./*.mac:./*.MAC:...`.
7. The assembler searches for macros or copy files first in uppercase, then in lowercase. The case of the extension is not changed. Mixed-case names are not recognized.
8. Either ASCII or EBCDIC files may be provided as input. The assembler will convert the ASCII input to EBCDIC for internal processing. The assembler determines if the data is in ASCII by looking for an ASCII blank (X'20'), ASCII asterisk (X'21') or ASCII numbers (X'30' to X'39') in the first record of the file.
   - Care must be exercised in using characters whose encodings are not stable when converted between EBCDIC and ASCII. Examples of such characters include not sign (¬), vertical bar (|), and square brackets ([ ]).
   - ASCII DBCS characters will not be correctly translated to EBCDIC DBCS.
9. TERM messages are written in ASCII. If you specify the TERM option but provide no `-t term_path_name`, output goes to stdout (the terminal), and can be redirected with `>` to a file.

To create ELF32 object files on zLinux, specify the HLASM ELF32 option (`--options='ELF32'`). You will also need to provide the following parameter so that the assembler can locate and load the object exit that does this translation:

```
-E /usr/bin/asmalib
```

For information about options specific to High Level Assembler on zLinux, especially the ELF32 option, and also ASMAXT2E messages, see the *HLASM Programmer's Guide*.

Chapter 19. Usage and limitations of High Level Assembler on zLinux

This chapter describes some points about the usage and limitations of High Level Assembler that you should consider when using the assembler on zLinux.

Usage

1. The generated listing is in EBCDIC, and will be difficult to view on zLinux. However, the Linux 'dd' command will automatically convert EBCDIC files to ASCII, insert newlines, and so on. The `'man'` pages for `'dd'` describe its options.
2. When you use FTP to copy source and macro files to zLinux, if you specify the 'binary' option, the file is transferred in EBCDIC. This may avoid unexpected mappings of EBCDIC characters to ASCII code points that HLASM does not recognize or process correctly.
3. Be careful if you edit files on zLinux, because the original 80-byte fixed-length structure of each record may not be retained.
4. Source files are recognized as EBCDIC or ASCII by checking the first record. While the test is normally reliable, there are some cases where the need for conversion is not correctly recognized. Also, some characters (such as those in C-type and G-type constants) may not have ASCII equivalents that will cause identical object code to be created.
5. High Level Assembler for zLinux does not use a work file and therefore the default size option has been set at 32M. This may be inadequate for large or complex assemblies, but can be increased by specifying the SIZE(xxM) option.
6. Transferring object or listing files to and from zLinux: if you wish to copy files to or from zLinux using FTP, set the `'binary'` and `'locsite fix 80'` options (for object files) or `'locsite fix 133'` (or 121), for listing files, before each `'get'`.

   For example, when sending the files to z/VM, set the mode to `'binary'` and issue the `'put'` FTP command. Then, you can post-process each file with either

   ```
   'PIPE <' fn ft fm '| deblock fixed '  80 '|> ' ofn oft ofm
   ```

   for object files, or

   ```
   'PIPE <' fn ft fm '| deblock fixed ' 133 '|> ' ofn oft ofm
   ```

   for listing files.

Limitations

1. High Level Assembler for zLinux executes in 32-bit addressing mode. When executing on zLinux systems running in 64-bit addressing mode, the Linux 32-bit compatibility interface is required for correct execution.
2. High Level Assembler for zLinux converts only OBJ object files to ELF32 format. To create ELF object files from GOFF object files, a separate conversion module is required.

The limitations that apply to the generation of ELF32 object files are described in the *HLASM Programmer's Guide*.

Chapter 20. Isolating the problem

This chapter covers the diagnostic process, diagnostic aids that are provided, and overcoming installation problems. You also use the information in this chapter to help you in diagnosing problems with the HLASM Toolkit Feature.

If a problem occurs while you are using High Level Assembler, or the Toolkit Feature, its cause might not be obvious. It might be an error in your program, in the assembler itself, or in some component of the assembler's operating environment. To help you identify the failure, use the following procedure.

Diagnosing the problem

This procedure gathers the diagnostic information required for developing a keyword string to search the software support database. It describes options that supply all available diagnostic information. You will need this information to discuss the problem with your IBM support representative if a search against the database fails to locate a fix for your problem.

1. Determine if the program has been changed since it was last assembled successfully. If it has, examine the changes. If the error is occurring in the changed code and cannot be corrected, note the change that caused the error. If possible, retain copies of both the original and the changed programs to submit with an Authorized Program Analysis Report (APAR) if it is required.
2. Ensure that you are assembling the correct version of the source code. You might have incorrectly identified the location of your source file. For example, check your data set names.
3. Determine if the problem looks like a wait or a loop. If it does, it might be a system problem. You should follow your installation's procedures for resolving such problems.
4. Follow the basic diagnostic procedures discussed in HLASM Programmer's Guide. If you receive an assembler error diagnostic message, you should verify the correct syntax and usage of the code that produced the error.
5. Correct all problems diagnosed by diagnostic messages, and make sure that previous diagnostic messages are not the cause of the current problem. Pay attention to warning messages (W-level messages). Identify the message by the following convention:
   - High Level Assembler messages are prefixed by the characters ASMA.
   - For messages with prefixes other than ASMA, the prefix determines which system/subsystem issued the message. Consult the applicable system/subsystem messages manual.
6. Your installation may have received an IBM Program Temporary Fix (PTF) for the problem. Make sure that all PTFs have been applied, so that your installation is at the latest maintenance level.
7. The Preventive Service Plan (PSP bucket), an online database available to IBM customers, gives information about product installation problems and other problems. Refer to the appropriate service planning section for your platform for more information:
   - "Preventive Service Planning" on page 6 for z/OS
   - "Preventive Service Planning" on page 41 for z/VM
   - "Preventive Service Planning" on page 82 for z/VSE

   For the relevant information for the HLASM Toolkit Feature, see the HLASM Toolkit Feature Installation and Customization Guide.
8. After the failure has been identified, consider writing a small test case that reproduces the problem. This test case should help you to:
   - Isolate the problem.
   - Distinguish between an error in the application program and an error in High Level Assembler.
   - Choose keywords that best describe the error.

9. Specify the following assembler options, in addition to the options originally specified, and reassemble the program. These options produce maximum diagnostic information that help you diagnose product errors. See theHLASM Programmer's Guide for more information about how to use these options.
   ```
   DXREF
   ESD
   FLAG(0,ALIGN,CONT,RECORD,SUBSTR)
   LIST(121)
   MXREF(FULL)
   PCONTROL(DATA,GEN,ON,MCALL,MSOURCE,UHEAD)
   RLD
   RXREF
   USING(MAP,WARN(15))
   XREF(FULL)
   ```
10. If the error symptoms change, return to step 4.
11. Record the sequence of events that led to the error condition. You might be able to use this information in developing a keyword string, and will need it if an APAR is required.
12. Begin developing the keyword string, using the procedure in Chapter 21, “Building a keyword string,” on page 117.

Diagnostic aids

In certain situations when an assembly cannot be completed, High Level Assembler requests its internal abnormal termination routine to produce a specially formatted dump. Diagnostic information in the dump can be useful if an APAR is needed. The types of data that can be extracted from the dump include:

- Assembly abnormal-termination messages
- Register contents when the abnormal termination was requested
- The assembler common-storage area
- The statement that was being processed when the abnormal termination was requested

The contents of the dump depend upon when the abnormal termination was requested during processing of the assembly.

This information should be retained and provided when requested by IBM.

Installation problems

You can avoid or solve most installation problems if you follow these steps:

1. Consult the PSP bucket (see step 7 on page 115).
2. Read any material that accompanies the installation tape.
3. Review the step-by-step installation procedure before installing High Level Assembler, or the HLASM Toolkit Feature.

If you still cannot solve the problem, develop a keyword string based on the symptoms of the problem as described in Chapter 21, “Building a keyword string,” on page 117.

Chapter 21. Building a keyword string

Failures in High Level Assembler can be described through the use of *keywords*. A keyword is a word or abbreviation assigned to describe one aspect of a product failure. A set of keywords, called a keyword string, can be used to describe the failure in detail. The procedures in this section will help you construct a keyword string that describes what you know about the product failure.

Information/Access or SoftwareXcel Extended can give you access to a computer-based abstract of the information in the software support database. This feature allows you to do your own search for previously recorded product failures before calling the IBM Support Center.

You may also find useful information at:

```
http://www.ibm.com/software/awdtools/hlasm/support.html
```

After it is constructed, the keyword string is used as a search argument to search against an IBM software support database, such as the Software Support Facility (SSF). The database contains keyword and text information describing all current problems, reported through APARs, and associated PTFs. IBM Support Center personnel have access to the software support database and are responsible for storing and retrieving the information. They use the keyword string to search the database and retrieve records that describe similar known problems.

If the keyword string produces a match in the software support database, the search might yield a fuller description of the problem and possibly identify a correction or circumvention. Such a search might yield several matches with previously reported problems. Review each error description carefully to determine if the problem description in the database matches your problem.

If a match is not found, use the keyword string you have constructed to describe the failure when contacting the IBM Support Center for assistance. Keywords ensure that identical program errors are described with identical keyword strings. Spelling the keywords exactly as they are presented in this book is especially important for a successful match.

Keyword usage

The first keyword in a keyword string identifies the failing component. The component identification for High Level Assembler is the product identifier 569623400. For HLASM Toolkit Feature this keyword is 569623401. A search of the software support database with this single keyword locates all problems reported for the whole assembler. Each additional keyword added to the keyword string narrows the scope of the search argument and helps to eliminate unnecessary examination of problem descriptions that have similar, but not matching, characteristics. In some cases, a correction for a product failure might be located with less than a full set of keywords. If you cannot follow the instructions for selecting a particular keyword, omit that keyword to avoid incorrectly identifying the problem. In general, if you contact IBM, you will be asked to identify your problem with a full set of keywords, as described here.

Figure 29 on page 118 shows the process of creating a keyword string. The keywords are indicated as you proceed through the diagram. A full set of keywords for High Level Assembler contains:

- The component identification
- The release level
- The type of failure
- One or more modifier keywords, depending on the type of failure, if applicable

Follow the steps in the keyword procedures until you are directed to the search argument procedure.

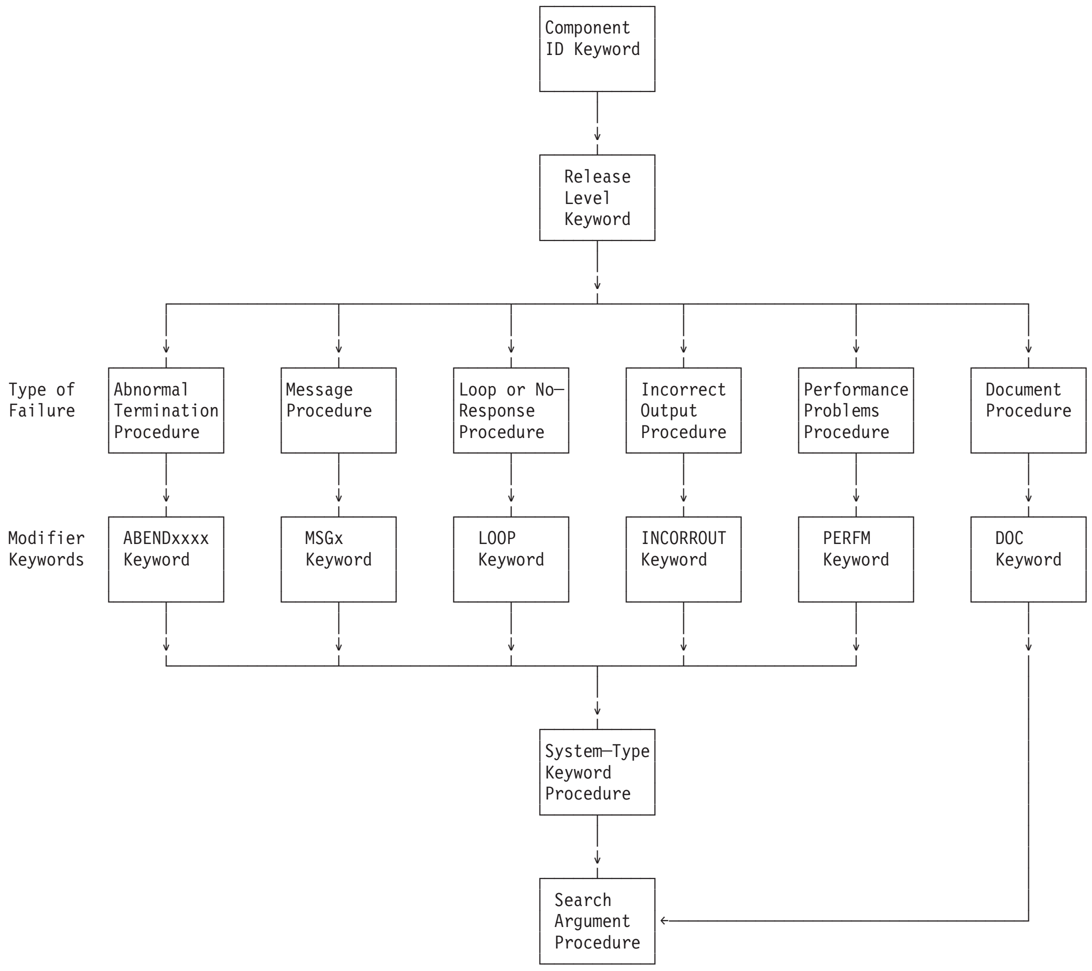


Figure 29. High Level Assembler—Problem Identification Using Keywords

Using the problem identification worksheet

You can use the worksheet described in “Problem identification worksheet” on page 125 to help you construct and record a keyword string. As you identify the keywords associated with your software problem, record them in the spaces provided.

Step 1. Component identification keyword procedure

This procedure shows what to specify in the component identification keyword. The component identification keyword is always the first keyword placed in the search argument string. It is derived from the program number and identifies the area within the software support database that contains APARs for High Level Assembler or the Toolkit Feature.

Use the component identification keyword with at least a type-of-failure keyword to search the software support database. If you use the component identifier keyword without additional keywords, a full listing of all APARs affecting High Level Assembler or the Toolkit Feature are produced.

Component Identification Procedure:

1. Use `569623400` as the component identification keyword. This number is the IBM High Level Assembler for z/OS & z/VM & z/VSE product identifier, 5696234, with a two-digit component number appended. The component number in this case is `00`. (The component number for the Toolkit Feature is `01`.)
2. If service updates have been applied to the licensed program, the update level of the last service update applied should be noted. (See your system programmer for the current service level of your High Level Assembler.) Although the service update level is not used as a keyword search argument, it is useful when reviewing APARs selected during the keyword search. The *Diagnostic Cross Reference and Assembler Summary* section of the assembler listing shows the last PTF applied to your High Level Assembler or Toolkit Feature.

Step 2. Release level keyword procedure

Use the following procedure to identify the specific release level of High Level Assembler under which you were operating when the failure occurred.

Release level procedure

1. Locate the release and modification level (denoted by `HLASM R`*r.m*) line at the top of the first page of your latest assembly output listing for the failing program. The release level line contains the current product identification data in the following format:

   ```
   HLASM Rr.m  yyyy/mm/dd hh.mm
   ```

   where *r.m* specifies the current release and modification number. The release level line for Release 6.0 of High Level Assembler is:

   ```
   HLASM R6.0  yyyy/mm/dd hh.mm
   ```
2. Specify the release level keyword, using the following format: *Rrm0*

   where *r* is the release level, and *m* is the modification level. The last character of this keyword is always zero.

   If Release 6.0 is the level you found in your listing, the release level keyword is R600.

   The following is an example of a set of keywords, consisting of the component identification and release level keywords:

   ```
   Component Identification:     569623400

   Release Level:                R600
   ```

Step 3. Type-of-failure keyword procedure

Various types of failures might occur in the High Level Assembler licensed program. Read the following table and select the type of failure that best describes the problem. Then go to the associated keyword procedure listed in this table for instructions on how to complete the keywords for that type of failure. If more than one keyword describes the problem you are experiencing, use the one that appears first in the table.

Table 48. Types of High Level Assembler failures

Type-of-Failure	Symptom	Procedure
Abnormal Termination	The assembler has ended abnormally without a message, or with a completion code that indicates a system abend has occurred.	"Abnormal termination problem procedures" on page 120.
Message Problems	A message indicates an assembler error, or there seems to be an error with the message itself.	"Message problem procedures" on page 121.

Table 48. Types of High Level Assembler failures (continued)

Type-of-Failure	Symptom	Procedure
No Response Problems	The assembler seems not to be doing anything, or is doing something repetitively.	"Loop or no response problem procedures" on page 122.
Output Problems	The output from the assembler is missing or incorrect.	"Output problem procedures" on page 122.
Performance Problems	The performance of the assembler is degraded.	"Performance problem procedures" on page 123.
Documentation Problems	Information in one of the High Level Assembler publications is incorrect or missing.	"Documentation problem procedures" on page 124.

Abnormal termination problem procedures

Use this procedure if High Level Assembler ends abnormally with a system message or a system abend code. Do not use this procedure if termination was accompanied by a message with the prefix ASMA. For those situations, see "Message problem procedures" on page 121. On z/OS, if High Level Assembler ends abnormally with a system abend code 322, 522, or 722, you should use the procedure for "Loop or no response problem procedures" on page 122.

ABENDxxxx procedure

1. On z/OS, if a system abend code is available, replace the *xxxx* of ABENDxxxx with the system abend code, prefixed by a zero. For example, if the code was a system abend 0C4, specify `ABEND00C4` as your keyword. If the failure occurred with a system abend 0C1, your set of keywords, so far, consists of:
   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              ABEND00C1
   ```
2. If you receive the following message on CMS, use `OPERATION` and `EXCEPTION` as keywords.
   ```
   DMSITP141T Operation exception occurred at 6BB40C in routine ASMAHL
   ```
   Your set of keywords then consists of:
   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              OPERATION EXCEPTION
   ```
3. On z/VSE, ABENDs will be accompanied by a message which describes the cause of the error. Use words from the message as keywords. For example, if you receive the following message, you could use `PROTECTION` and `EXCEPTION` as keywords.
   ```
   0S03I PROGRAM CHECK INTERRUPTION - HEX LOCATION 0006D2D0 - CONDITION CODE 0
   - PROTECTION EXCEPTION
   ```
   Your set of keywords then consists of:
   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              PROTECTION EXCEPTION
   ```
4. On CMS or z/VSE, if using words from the message as your type-of-failure keyword does not produce a match in the software support database, use the equivalent z/OS system abend code as the type-of-failure keyword.

   The most common errors, with their z/OS system abend codes, are:

Table 49. System Abend Codes

Error	System Abend Code
Operation exception	0C1
Privileged operation exception	0C2
Execute exception	0C3
Protection exception	0C4
Addressing exception	0C5
Specification exception	0C6
Data exception	0C7
Fixed-point overflow exception	0C8
Fixed-point divide exception	0C9

5. Determine with which assembler options the failure occurs. If the failure occurs only when using certain options, indicate those options in the keyword string. Select the applicable modifier keyword from the list shown in "Assembler options" on page 135.
6. Continue with "Step 4. System-type keyword" on page 125.

Message problem procedures

The following message types are issued by High Level Assembler:
- *ASMAHL* command errors (CMS)
- Assembler-error diagnostic messages
- Assembly abnormal-termination messages

High Level Assembler messages are identified by the prefix ASMA.

On CMS, messages issued by the *ASMAHL* command have a prefix of ASMACMS.

The format of the message identifier is ASMA*nnnc*, where:

ASMA
is the message prefix identifying all High Level Assembler messages.

nnn is the message number.

c is one of the following message severity characters:

I for informational messages

N for notification messages

W for warning messages

E for normal error messages

S for severe error condition messages

C for critical error condition messages

U for unrecoverable error condition messages

Assembler error diagnostic messages are numbered `ASMA001` to `ASMA899`. Assembly abnormal termination messages are numbered `ASMA900` to `ASMA999`. Messages with other prefixes are issued by operating systems, subsystems, and access methods. They should not be addressed as High Level Assembler product problems. See the messages manuals for the relevant components.

Use the MSGxxx keyword procedure for any one of the following conditions:
- A message is issued under a set of conditions that should not have caused it to be issued.
- A message contains incorrect data or is missing data.

- A message indicates an internal assembler error (for example, ASMA951).

Do not use this procedure if the assembler ended with a system abend code, or a system message indicating an abnormal-termination. In these cases, use the "Abnormal termination problem procedures" on page 120.

MSGxxx procedure

1. Replace the *xxx* of MSGxxx with the complete message identifier, but do not include the severity character (if any). For example, if assembly abnormal termination message ASMA950U is issued, the MSGxxx keyword is MSGASMA950. Your set of keywords, so far, consists of:
   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              MSGASMA950
   ```
2. Determine with which assembler options the failure occurs. If the failure occurs only when using certain options, indicate those options in the keyword string. Select the applicable modifier keyword from the list shown in "Assembler options" on page 135.
3. Continue with "Step 4. System-type keyword" on page 125.

Loop or no response problem procedures

Use the LOOP keyword procedure for any of the following conditions:

- A program seems to be doing nothing or is doing something repetitively.
- A job does not reach completion.
- On z/OS, the system abend code is 322, 522, or 722, which means that your program has timed out or exceeded its output limits.

If the program appears to be in a WAIT state, follow your local procedures for resolution.

LOOP procedure

1. Determine with which assembler options the failure occurs. If the failure occurs only when using certain options, indicate those options in the keyword string. Select the applicable modifier keyword from the list shown in "Assembler options" on page 135.
2. If you are running on z/OS and the error is a system abend, with a system abend code indicating not enough time, or inadequate output limits, increase the allotment and rerun your program. If the problem is still unresolved, your set of keywords, so far, consists of:
   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              LOOP
   ```
3. Continue with "Step 4. System-type keyword" on page 125.

Output problem procedures

Use this procedure when the output appears to be incorrect or missing, but the program otherwise ended normally.

INCORROUT procedure

1. If the data or records were repeated endlessly, use the "LOOP procedure" instead of the INCORROUT procedure to create your keyword string.
2. Use INCORROUT as your type-of-failure keyword.
3. If the error was detected because of incorrect or missing output from an assembly that otherwise completed successfully, select a modifier keyword from the following table to describe the type of error in the output.

Modifier Keyword	Type of Incorrect Output
DUPLICATE	Some data or records were duplicated, but were not repeated endlessly.
INVALID	The output that appeared was incorrect or not as expected.
MISSING	Some expected output was missing.

4. Select another modifier keyword from the following table to describe the portion of the output in which the error occurred.

Modifier Keyword	Portion of Output in Error
ADATA	Associated data file.
DXREF	DSECT cross reference.
ESD	External symbol dictionary listing. If the external symbol dictionary part of the object program is in error, use the OBJECT keyword followed by the ESD keyword.
GOFF (z/OS and CMS)	Machine-language generalized object program.
MESSAGE	Diagnostic message.
MXREF	Macro and copy code source summary, and macro and copy code cross reference.
OBJECT	Machine-language object program.
RLD	Relocation Dictionary listing. If the relocation dictionary part of the object program is in error, use the OBJECT keyword followed by the RLD keyword.
RXREF	General Purpose Register Cross Reference
SOURCE	Source listing.
STAT	Statistics and error listing.
TERM	Progress and diagnostic messages on SYSTERM data set for z/OS, on the terminal for CMS, or on SYSLOG for z/VSE.
UMAP	USING map.
XREF	Ordinary Symbol and Literal Cross Reference listing.

5. Determine with which assembler options the failure occurs. If the failure occurs only when using certain options, indicate those options in the keyword string. Select the applicable modifier keyword from the list shown in "Assembler options" on page 135.

   For example, if you think that the assembler has given an incorrect Ordinary Symbol and Literal Cross Reference section of the listing when assembling with the XREF(FULL) option, your set of keywords, so far, consists of:

```
Component Identification:    569623400

Release Level:               R600

Type of Failure:             INCORROUT

Modifiers:                   INVALID
                             XREF
                             FULL
```

6. Continue with "Step 4. System-type keyword" on page 125.

Performance problem procedures

Most performance problems can be related to system tuning and should be handled by system engineers and system programmers. Use the PERFM keyword when the performance problem cannot be corrected by system tuning and performance is below expectations as documented in an IBM product publication.

PERFM procedure

1. Record the actual and expected performance measurements for your system configuration. Note the order number and page of the IBM document that is the source of your performance expectations. You will be asked for this information if you contact the IBM Support Center. If you prepare materials for an APAR, you should also include this information in the error description.
2. Determine with which assembler options the failure occurs. If the failure occurs only when using certain options, indicate those options in the keyword string. Select the applicable modifier keyword from the list shown in “Assembler options” on page 135.
3. Use PERFM as your type-of-failure keyword. For example, your set of keywords for performance problems could consist of:

   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              PERFM
   ```
4. Continue with “Step 4. System-type keyword” on page 125.

Documentation problem procedures

Use the DOC keyword procedure when you notice a problem caused by incorrect or missing information in one of the published High Level Assembler documents.

DOC procedure

1. Locate the page or pages in the document where the problem occurs, and prepare a description of the error and the problem it caused. This information is required for APAR preparation if no similar problem is found in the software support database.
2. Decide whether this documentation problem is severe enough to cause lost time for other users.

   If the problem is not severe, follow the instructions in “How to send your comments to IBM” on page xiv. Include the problem description you have developed, along with your name and return address, so that IBM can respond to your comments.

   If the problem is severe enough to cause lost time for other users, continue creating your keyword string to determine whether IBM has a record of the problem. If this is a new problem, you will be asked to submit a severity-3 or severity-4 documentation (DOC) APAR.
3. Use the order number on the cover of the document together with the DOC keyword as your type-of-failure keyword, but omit the hyphens. Leave a single space between DOC and the document number. The number following the last hyphen in the document number indicates the document release level. If the document release level number has only one digit, it must be preceded by a zero. For example if the order number is SC26-4940-05 (*HLASM Language Reference*), use SC26494005. Your set of keywords consists of:

   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              DOC SC26494005
   ```
4. If, after searching the IBM software support database, you do not find a matching description, you may want to search again, using the following format:

   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              DOC SC264940**
   ```

   The two asterisks appended to the document number cause a search for all problems reported for the document rather than only those for a specific release of the document.
5. Continue with “Step 4. System-type keyword” on page 125.

Step 4. System-type keyword

Use this procedure to indicate which system you were operating on when High Level Assembler failed.

System-type procedure

1. If the failure occurred while your program was assembling:
   - On z/OS, use MVS or z/OS as your system-type keyword.
   - On CMS, use CMS as your system-type keyword.
   - On z/VSE, use VSE or z/VSE as your system-type keyword.
2. Use ESA as your next system-type keyword.

   For example, the keywords you use when you have received the assembly abnormal termination message, ASMA950U, consists of:

   ```
   Component Identification:     569623400

   Release Level:                R600

   Type of Failure:              MSGASMA950

   System Type:                  MVS ESA
   ```

Problem identification worksheet

Component Identification:

____

Release Level:

____

Type of Failure:

____

System Type:

____

Modifiers:

____

____

____

____

Some keywords may not be applicable to all software problems. See Chapter 24, “Modifier keywords,” on page 133 for information on the modifiers you can specify.

Chapter 22. Using the keyword string as a search argument

This chapter explains how to use the keyword string you have developed to search the software support database. You can conduct the search yourself if you have access to the correct database, or you can request that IBM conduct the search.

How to use the keyword string

Searches against a software support database will be most successful if you follow these rules:

- Use only the keywords given in this book.
- Spell keywords the way they are spelled in this book. Any variation in spelling may result in an unsuccessful search.
- Include all the applicable keywords in any discussion with IBM support personnel or in an APAR.

Search argument procedure

1. Search the software support database, using the full set of keywords you have developed. For example, given the following list:

   ```
   Component Identification:    569623400

   Release Level:               R600

   Type of Failure:             ABEND00C4

   System Type:                 MVS ESA

   Modifiers:                   MXREF
   ```

   your keyword string consists of:

   ```
   569623400 R600 ABEND00C4 MVS ESA MXREF
   ```

2. If the search produces a list of APARs, continue with step 3, otherwise go to step 6.
3. When your search is complete, eliminate from the list of possible APAR fixes those that have already been applied to your system.
4. Compare each of the remaining closed APAR descriptions with the current failure symptoms.
5. If a match is found, find out if there is a corresponding PTF. If necessary, you can order the PTF from the IBM Support Center. If there is a PTF, apply it to your system and exit this procedure.

   For information about how to apply a PTF, refer to the cover letter for the PTF to be applied.
6. If the search **did not** produce a list of APARs, or an APAR description matching the current failure is not found, expand the search by using the following techniques:
   a. Omit the release level keyword (for example, `R600`) from the search. This expands the search to include similar failures on other release levels.
   b. Drop one keyword from the right end of the search argument string. The diagnostic procedures directed you to construct the keyword string with the most significant keywords listed first. By dropping a keyword from the right, you eliminate the least significant keyword, thus expanding your search while maintaining the relevancy of your search argument string. Perform the search against the software support data base using your shortened search argument string. Repeat this step as necessary.
7. If a match is not found using the preceding techniques, go to Chapter 23, "Preparing an APAR," on page 129.

Chapter 23. Preparing an APAR

This chapter explains how to prepare an Authorized Program Analysis Report (APAR) if you are asked to do so by IBM support personnel.

You may be asked to prepare an APAR if:
- You have eliminated user errors as a possible cause of the problem.
- You have followed the diagnostic procedures presented in this book.
- The keyword search has proved unsuccessful.

Initiating an APAR

1. Contact the IBM Support Center for assistance. Tell the support personnel that you have used this manual to create a keyword string. Be prepared to supply the following information:
   - Customer number
   - Operating system
   - Operating system release level
   - Current High Level Assembler maintenance level (PTF list and list of APAR fixes applied)
   - The various keyword strings used to search the software support data base
   - Processor serial and model number
2. From the following list, you might be asked to include the applicable High Level Assembler environmental information with your APAR:
   - Job control statements
   - Any spooled CMS consoles or special EXECs
   - The following assembler listings:
     - High Level Assembler Options Summary
     - External Symbol Dictionary
     - Source and Object
     - Relocation Dictionary
     - Ordinary Symbol and Literal Cross Reference
     - Unreferenced Symbols in a CSECT
     - DSECT Cross Reference
     - Macro and Copy Code Source Summary
     - Macro and Copy Code Cross Reference
     - General Purpose Register Cross Reference
     - Diagnostic Cross Reference and Assembler Summary
     - USING map
   - Machine-readable copy of the program causing the problem, including all macros and copy members required by the program. This should be the smallest, least complex form of the program that still produces the error.
   - SYSADATA (z/OS and CMS) or SYSADAT (z/VSE) output

In addition, the console log might be helpful in reproducing the error. Any listings supplied must be from the High Level Assembler assembly that failed.

You may also be asked to provide trace information from the assembly that failed.

Table 50 provides the options or methods used to produce the documentation required. Many of these materials may have been previously produced in the required format during the development of the keyword string. (See "Diagnosing the problem" on page 115.) Any additional requirements are explained after the table.

Table 50. Problem resolution documentation descriptions

Item	Materials Required	How to Obtain Materials
1	Machine-readable source program	
2	Assembly listings:	
	High Level Assembler Options Summary	by default
	External Symbol Dictionary	ESD option
	Source and Object	LIST option
	Relocation Dictionary	RLD option
	Ordinary Symbol and Literal Cross Reference	XREF(SHORT) or XREF(FULL)
	Unreferenced Symbols Defined in a CSECT	XREF(UNREFS)
	DSECT Cross Reference	DXREF option
	Macro and Copy Code Source Summary	MXREF(FULL) option or MXREF(SOURCE)
	Macro and Copy Code Cross Reference	MXREF(FULL) option or MXREF(XREF)
	General Purpose Register Cross Reference	RXREF option
	USING map listing	USING(MAP) option
	Diagnostic Cross Reference and Assembler Summary	by default
	Assembler abnormal termination dump	by default
3	Assembler system ABEND dump	On z/OS: SYSUDUMP DD statement (as directed by IBM support personnel). On CMS: VM DUMP command (as directed by IBM support personnel). On z/VSE: JCL OPTION DUMP or PARTDUMP (as directed by IBM support personnel).
4	Partition/region size/virtual storage size	JCL or system programmer.
5	List of applied PTFs	System programmer.
6	z/OS job control statements with MSGLEVEL(1,1), TSO ALLOCATE statements, CMS interactive session listing, or z/VSE job control statements with OPTION LOG.	See "Job control statements (z/OS)" on page 131, "Interactive environment (CMS)" on page 131 or "Job control statements (z/VSE)" on page 131.
7	SYSADATA (z/OS and CMS) or SYSADAT (z/VSE) output	See "SYSADATA (z/OS and CMS) or SYSADAT (z/VSE) output" on page 131.
8	Trace Output (ON REQUEST)	See Chapter 25, "Internal Trace Facility," on page 141.

Job control statements (z/OS)

- Supply the JCL listings used to run the assembly, including an expanded list of the cataloged procedures used.

Interactive environment (CMS)

- The listing supplied must include all parts of the interactive session that dealt with this problem.
- Supply full details of the interactive environment immediately before you invoked the assembler.

  Use the CP SPOOL command to spool your console for printing, then issue the following commands:

  ```
  QUERY DISK *
  QUERY FILEDEF
  QUERY INPUT
  QUERY LIBRARY
  QUERY MACLIB
  QUERY OUTPUT
  QUERY SEARCH
  QUERY SET
  QUERY TERMINAL
  QUERY VIRTUAL
  ```

  This provides the necessary details of your interactive environment.

Job control statements (z/VSE)

- Supply the JCL listings used to run the assembly, including an expanded list of the cataloged procedures used.
- Run LSERV to list the system standard labels and partition standard labels in effect.
- Issue a LISTIO for the partition to list the system logical unit assignments active.
- Issue a MAP command on the z/VSE console to show the partition size and GETVIS storage allocation.
- Run LIBR with the command LISTDIR SDL to list the contents of the system directory list (phases that have been placed in the SVA).

SYSADATA (z/OS and CMS) or SYSADAT (z/VSE) output

If requested, supply any SYSADATA output data.

Submitting the APAR documentation

IBM support personnel will tell you how you should submit the material for an APAR to IBM. This may be by email, or using an FTP process as described at:

`http://www.ibm.com/de/support/ecurep/`

Chapter 24. Modifier keywords

One or more modifier keywords may be used in the same keyword string to define the problem. Additional modifiers help make the search argument more specific. Use the capitalized spelling of the modifier in the keyword string. The various types of modifier keywords listed below are:

- Assembler Language
  - Ordinary assembler instructions
  - Conditional assembly instructions
  - Macro processing instructions
  - System variable symbols
  - Machine instructions

  See "High Level Assembler Language Elements" for a list of the assembler language elements and the associated keywords to be used in the keyword string.
- Assembler Options

  Select from your assembly listing those assembler options that you consider significant to the type of failure. See "Assembler options" on page 135 for a list of the assembler options. The option name itself (and suboption if applicable) is the keyword.

High Level Assembler Language Elements

LANGUAGE ELEMENT	**KEYWORD**
ACONTROL	ACONTROL
ACTR	ACTR
ADATA	ADATA
AGO	AGO
AEJECT	AEJECT
AIF	AIF
AINSERT	AINSERT
ALIAS	ALIAS
AMODE	AMODE
ANOP	ANOP
AREAD	AREAD
ASPACE	ASPACE
'Built-in Function'[1]	
CCW	CCW
CCW0	CCW0
CCW1	CCW1
CEJECT	CEJECT
CATTR	CATTR(z/OS and CMS)
CNOP	CNOP
COM	COM
COPY	COPY
CSECT	CSECT
CXD	CXD
DC	DC
DROP	DROP
DS	DS
DSECT	DSECT
DXD	DXD
EJECT	EJECT
END	END

LANGUAGE ELEMENT	KEYWORD
ENTRY	ENTRY
EQU	EQU
EXITCTL	EXITCTL
EXTRN	EXTRN
GBLA	GBLA
GBLB	GBLB
GBLC	GBLC
ICTL	ICTL
LCLA	LCLA
LCLB	LCLB
LCLC	LCLC
LOCTR	LOCTR
LTORG	LTORG
'Machine instruction'[2]	
MEXIT	MEXIT
MHELP	MHELP
MNOTE	MNOTE
OPSYN	OPSYN
ORG	ORG
POP	POP
PRINT	PRINT
PUSH	PUSH
REPRO	REPRO
RMODE	RMODE
RSECT	RSECT
SETA	SETA
SETAF	SETAF
SETB	SETB
SETC	SETC
SETCF	SETCF
SPACE	SPACE
START	START
&SYSADATA_DSN	SYSADATA DSN
&SYSADATA_MEMBER	SYSADATA MEMBER
&SYSADATA_VOLUME	SYSADATA VOLUME
&SYSASM	SYSASM
&SYSCLOCK	SYSCLOCK
&SYSCODEPAGE	SYSCODEPAGE
&SYSDATC	SYSDATC
&SYSDATE	SYSDATE
&SYSECT	SYSECT
&SYSIN_DSN	SYSIN DSN
&SYSIN_MEMBER	SYSIN MEMBER
&SYSIN_VOLUME	SYSIN VOLUME
&SYSJOB	SYSJOB
&SYSLIB_DSN	SYSLIB DSN
&SYSLIB_MEMBER	SYSLIB MEMBER
&SYSLIB_VOLUME	SYSLIB VOLUME
&SYSLIN_DSN	SYSLIN DSN
&SYSLIN_MEMBER	SYSLIN MEMBER
&SYSLIN_VOLUME	SYSLIN VOLUME
&SYSLIST	SYSLIST
&SYSLOC	SYSLOC
&SYSMAC	SYSMAC

LANGUAGE ELEMENT	KEYWORD
&SYSM_HSEV	SYSM HSEV
&SYSM_SEV	SYSM SEV
&SYSNEST	SYSNEST
&SYSNDX	SYSNDX
&SYSOPT_DBCS	SYSOPT DBCS
&SYSOPT_OPTABLE	SYSOPT OPTABLE
&SYSOPT_RENT	SYSOPT RENT
&SYSOPT_XOBJECT	SYSOPT XOBJECT
&SYSPARM	SYSPARM
&SYSPRINT_DSN	SYSPRINT DSN
&SYSPRINT_MEMBER	SYSPRINT MEMBER
&SYSPRINT_VOLUME	SYSPRINT VOLUME
&SYSPUNCH_DSN	SYSPUNCH DSN
&SYSPUNCH_MEMBER	SYSPUNCH MEMBER
&SYSPUNCH_VOLUME	SYSPUNCH VOLUME
&SYSSEQF	SYSSEQF
&SYSSTEP	SYSSTEP
&SYSSTMT	SYSSTMT
&SYSTEM_ID	SYSTEM ID
&SYSTERM_DSN	SYSTERM DSN
&SYSTERM_MEMBER	SYSTERM MEMBER
&SYSTERM_VOLUME	SYSTERM VOLUME
&SYSTIME	SYSTIME
&SYSVER	SYSVER
TITLE	TITLE
USING	USING
WXTRN	WXTRN
XATTR	XATTR(z/OS and CMS)

1. For a conditional assembly language built-in function you should use the two keywords "BIF" and the built-in function name.
2. For a machine instruction language element, you should use the machine instruction mnemonic as the keyword.

Assembler options

The option name (and suboption if applicable) is the keyword.

ASSEMBLER OPTION
ADATA
ALIGN
ASA
BATCH
CODEPAGE
COMPAT
COMPAT CASE
COMPAT LITTYPE
COMPAT MACROCASE
COMPAT SYSLIST
COMPAT NOCASE
COMPAT NOLITTYPE
COMPAT NOMACROCASE
COMPAT NOSYSLIST
DBCS
DECK (z/OS and CMS)
DELETE

ASSEMBLER OPTION
DSECT
DISK (CMS)
DXREF
ERASE (CMS)
EXIT
EXIT ADEXIT
EXIT INEXIT
EXIT TRMEXIT
EXIT LIBEXIT
EXIT PRTEXIT
EXIT OBJEXIT
ESD
FLAG n
FLAG ALIGN
FLAG CONT
FLAG EXLITW
FLAG IMPLEN
FLAG PAGE0
FLAG PUSH
FLAG RECORD
FLAG SUBSTR
FLAG USING0
FLAG NOALIGN
FLAG NOCONT
FLAG NOEXLITW
FLAG NOIMPLEN
FLAG NOPAGE0
FLAG NOPUSH
FLAG NORECORD
FLAG NOSUBSTR
FLAG NOUSING0
FOLD
GOFF (z/OS and CMS)
GOFF ADATA (z/OS and CMS)
GOFF NOADATA (z/OS and CMS)
INFO
LANGUAGE
LANGUAGE DE
LANGUAGE EN
LANGUAGE ES
LANGUAGE EU
LANGUAGE JP
LIBMAC
LINECOUNT
LIST
LIST 121 (z/OS and CMS)
LIST 133 (z/OS and CMS)
LIST MAX (z/OS and CMS)
LIST YES (z/OS and CMS)
MACHINE S370
MACHINE S370XA
MACHINE S370ESA
MACHINE S390
MACHINE S390E

ASSEMBLER OPTION
MACHINE ZSERIES
MACHINE ZS
MACHINE ZSERIES-2
MACHINE ZS-2
MACHINE ZSERIES-3
MACHINE ZS-3
MACHINE ZSERIES-4
MACHINE ZS-4
| MACHINE ZSERIES-5
| MACHINE ZS-5
| MACHINE ZSERIES-6
| MACHINE ZS-6
MACHINELIST
MXREF
MXREF FULL
MXREF SOURCE
MXREF XREF
NOADATA
NOALIGN
NOASA (z/OS and CMS)
NOBATCH
NOCOMPAT
NODBCS
NODECK (z/OS and CMS)
NODXREF
NOERASE (CMS)
NOEXIT
NOESD
NOFOLD
NOGOFF (z/OS and CMS)
NOINFO
NOLIBMAC
NOLIST
NOMACHINELIST
NOMXREF
NOOBJECT
NOOPTABLELIST
NOPCONTROL
NOPESTOP
NOPRINT (CMS)
NOPROFILE
NORA2
NORENT
NORLD
NORXREF
NOSEG (CMS)
NOSUPRWARN
NOTERM
NOTEST
NOTHREAD
NOTRANSLATE
NOTYPECHECK
NOUSING
NOWORKFILE

ASSEMBLER OPTION
NOXOBJECT (z/OS and CMS)
NOXREF
OBJECT
OPTABLE DOS
OPTABLE ESA
OPTABLE UNI
OPTABLE XA
OPTABLE 370
OPTABLE ZOP
OPTABLE YOP
OPTABLE ZS3
OPTABLE ZS4
| OPTABLE ZS5
| OPTABLE ZS6
OPTABLELIST
PCONTROL
PCONTROL ON
PCONTROL OFF
PCONTROL GEN
PCONTROL DATA
PCONTROL UHEAD
PCONTROL MCALL
PCONTROL MSOURCE
PCONTROL NOGEN
PCONTROL NODATA
PCONTROL NOUHEAD
PCONTROL NOMCALL
PCONTROL NOMSOURCE
PESTOP
PRINT (CMS)
PROFILE
RA2
RENT
RLD
RXREF
SECTALGN
SEG (CMS)
SIZE
SIZE MAX
SIZE MAX,ABOVE
SUPRWARN
SYSPARM
TERM
TERM WIDE
TERM NARROW
TEST
THREAD
TRANSLATE
TYPECHECK
TYPECHECK MAGNITUDE
TYPECHECK NOMAGNITUDE
TYPECHECK REGISTER
TYPECHECK NOREGISTER
USING

ASSEMBLER OPTION
USING LIMIT
USING NOLIMIT
USING MAP
USING NOMAP
USING WARN
USING NOWARN
WORKFILE
XOBJECT (z/OS and CMS)
XOBJECT ADATA (z/OS and CMS)
XOBJECT NOADATA (z/OS and CMS)
XREF
XREF SHORT
XREF FULL
XREF UNREFS

Chapter 25. Internal Trace Facility

The Internal Trace Facility (ITF) provides the IBM support personnel with information to assist with debugging errors in High Level Assembler.

You might be asked to provide an internal trace, or assist IBM support personnel in obtaining an internal trace, if the information described in Table 50 on page 130 is not sufficient to resolve the reported problem.

How to invoke the Internal Trace Facility

To invoke the Internal Trace Facility the following procedures are required:

1. IBM provides a trace control module which must be included in a load library (on z/OS and CMS), in the standard load module search order, or in a library in the standard phase search order (on z/VSE), that is available when the assembly is run. Your IBM support representative will provide you with the necessary instructions to perform this procedure.
2. Run the assembly with the ITF assembler option. The ITF option syntax is described below under "ITF assembler option."
3. Your IBM support representative will advise you of the best method for providing this material, and of sending it to IBM.

Specifying the trace data set

Under z/OS: use the SYSTRACE DD statement to define the Internal Trace Facility output data set:

```
//SYSTRACE DD DSN=dsname,DISP=(NEW,CATLG),SPACE=(CYL,(primary,secondary)),
//         DCB=(LRECL=81,BLKSIZE=n*81,RECFM=F)
```

Under CMS: issue the SYSTRACE FILEDEF command before the High Level Assembler command, to specify the output trace data set. For example, you could specify:

```
FILEDEF SYSTRACE DISK fn SYSTRACE m1 (RECFM F LRECL 81
```

If you do not do this, High Level Assembler will issue this FILEDEF command for you.

Under z/VSE: use the following JCL to specify the output trace data set:

```
// ASSGN SYSTRAC,DISK,VOL=volser,SHR
// DLBL SYSTRAC,'dsname',0,SD
// EXTENT SYSTRAC,volser,1,0,start,tracks
```

CAUTION:
The trace output data set can be large depending on the information requested by IBM.

ITF assembler option

Invoke the Internal Trace Facility by specifying the ITF assembler option.

```
►►──ITF(xx)──────────────────────────────────────────────────────────────►◄
```

Where *xx* specifies the suffix of the trace module, as supplied by IBM.

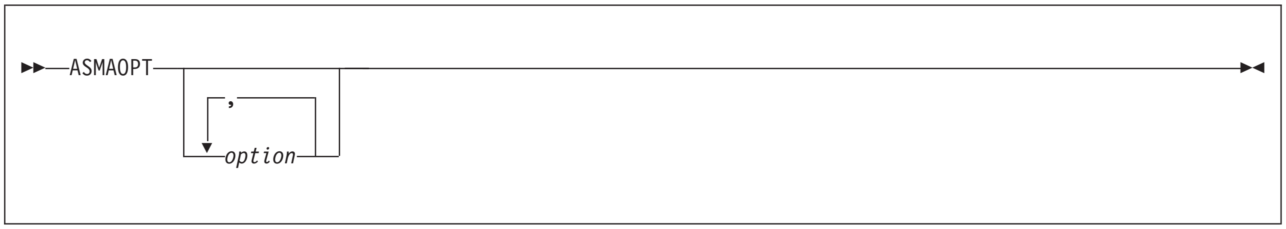
Appendix A. High Level Assembler Options

This appendix lists the options for High Level Assembler. In particular, the IBM-supplied default value for each option is indicated. You can use the ASMAOPT installation macro to select different values to be the defaults for your site.

For more information about how to change the default options, using ASMAOPT, see “Step 3: Change default options and DDNAMES” on page 27 (z/OS), “Changing option and DDNAMES defaults” on page 63 (CMS), or “Changing default options” on page 97 (z/VSE).

ASMAOPT

The installation macro, ASMAOPT, lets you specify installation defaults for assembler options during installation of High Level Assembler.

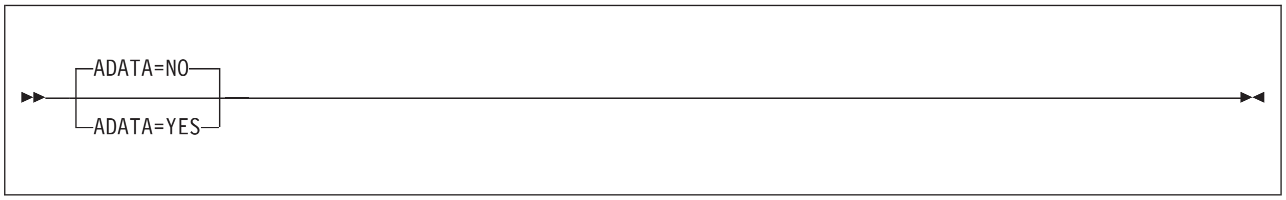


Defaults
: If an option is not specified when the ASMAOPT macro is assembled, then the IBM-supplied default is installed for that option. The default value is listed with each option.

ASMAOPT options

The operands of the ASMAOPT macro, and the values that can specified for each operand, are described below.

ADATA



YES
: associated data is written to the file defined by the SYSADATA DD statement (z/OS), the FILEDEF SYSADATA command (CMS), or the DLBL SYSADAT statement (z/VSE).

NO no associated data is collected.

Default
: ADATA=NO

ADEXIT

```
►►──ADEXIT=──┬────────────────────────┬──────────────────────────────►◄
             └─(name─┬─────────┬─)─┘
                     └─,string─┘
```

name
: identifies the name of a module that is loaded and called by the assembler to monitor the associated data records written by the assembler to SYSADATA (z/OS and CMS), or SYSADAT (z/VSE).

string
: the character string that is passed to the exit module as part of the parameter list built by the assembler. The character string is up to 64 characters in length. Any character can be included in the string, subject to the rules for building character strings defined in *HLASM Language Reference*. If the string includes blanks, commas, or parentheses, enclose it in apostrophes.

Default
: No exit specified.

Note: This option can be specified as an assembler invocation parameter by specifying the EXIT(ADEXIT(mod5(str5))) option. For more information, see the *HLASM Programmer's Guide*.

ALIGN

```
      ┌─ALIGN=YES─┐
►►────┼───────────┼──────────────────────────────────────────────────►◄
      └─ALIGN=NO──┘
```

YES
: instructs the assembler to check alignment of addresses in machine instructions for consistency with the requirements of the operation code type. DC, DS, DXD, and CXD are aligned on the correct boundaries.

NO
: instructs the assembler not to check alignment of unprivileged machine instruction data references, but still to check instruction references and privileged machine instruction data references. DC, DS, and DXD are aligned on the correct boundaries only if the duplication factor is 0.

Default
: ALIGN=YES

See also "ALIGNWARN" on page 145.

ALIGNWARN

```
   ┌─ALIGNWARN=YES─┐
►►─┤               ├──────────────────────────────────────────────────►◄
   └─ALIGNWARN=NO──┘
```

YES
instructs the assembler to issue one of the messages ASMA033I, ASMA212W, or ASMA213W when there is an alignment error.

NO instructs the assembler not to issue a message when there is an alignment error.

Default
ALIGNWARN=YES

See also “ALIGN” on page 144.

Note: This option can be specified as an assembler invocation parameter by specifying the FLAG(ALIGN) option.

ASA (z/OS and CMS)

```
►►──ASA=──┬─────┬─────────────────────────────────────────────────────►◄
          ├─NO──┤
          └─YES─┘
```

NO instructs the assembler to use machine printer-control characters in records written to the assembler listing file.

YES
instructs the assembler to use American National Standard printer-control characters in records written to the assembler listing file.

BATCH

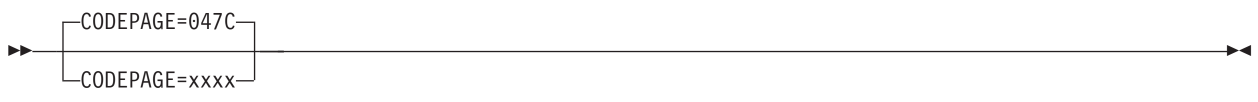
```
   ┌─BATCH=YES─┐
►►─┤           ├──────────────────────────────────────────────────────►◄
   └─BATCH=NO──┘
```

YES
instructs the assembler that multiple assembler source programs might be in the input file. The first statement of the second and subsequent source programs must immediately follow the END statement of the previous source program. An end-of-file must immediately follow the last source program.

NO instructs the assembler that only one assembler source program is in the input file. Statements after the END statement are ignored.

Default
BATCH=YES

CODEPAGE

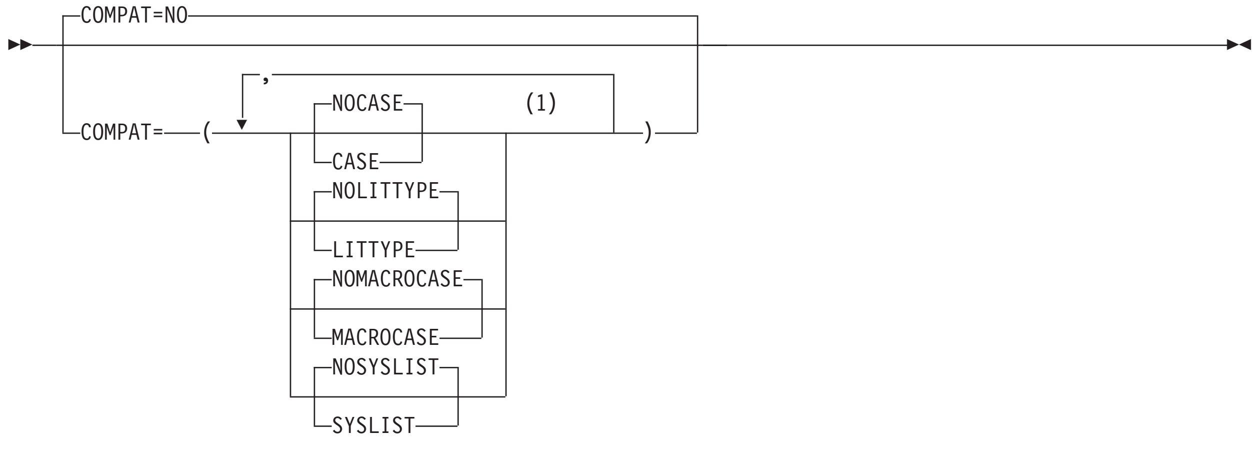


047C
Specifies that characters contained in the Unicode character (CU-type) data constants (DCs) are to be converted using the ECECP: International 1 Unicode-3 mappings contained in module ASMA047C.

xxxx
Specifies that characters contained in the Unicode character (CU-type) data constants (DCs) are to be converted using the Unicode mapping table module ASMA*xxxx* where *xxxx* is the hexadecimal value of the number of the code page contained in the module.

Default
CODEPAGE=047C

COMPAT



Notes:

1 Choose at least one option.

CASE
Instructs the assembler to maintain uppercase alphabetic character set compatibility with earlier assemblers. It restricts language elements to uppercase alphabetic characters *A* through *Z* if they were so restricted in earlier assemblers.

NOCASE
Instructs the assembler to allow a mixed-case alphabetic character set.

LITTYPE
Instructs the assembler to return "U" as the type attribute for all literals.

NOLITTYPE
Instructs the assembler to provide the correct type attribute for literals once they have been defined.

MACROCASE
Instructs the assembler to convert lowercase alphabetic characters (*a* through *z*) in unquoted macro operands to uppercase alphabetic characters (*A* through *Z*).

NOMACROCASE
Instructs the assembler not to convert lowercase alphabetic characters (*a* through *z*) in unquoted macro operands.

SYSLIST
Instructs the assembler to treat sublists in SETC symbols as compatible with earlier assemblers. SETC symbols that are assigned parenthesized sublists are treated as character strings, not sublists, when passed to a macro definition in an operand of a macro instruction.

NOSYSLIST
Instructs the assembler not to treat sublists in SETC symbols as character strings, when passed to a macro definition in an operand of a macro instruction.

NO Instructs the assembler to allow lowercase alphabetic characters *a* through *z* in all language elements, to treat sublists in SETC symbols as sublists when passed to a macro definition in the operand of a macro instruction, and to provide the correct type attribute for literals once they have been defined.

Default
COMPAT=NO

CONTWARN

```
     ┌─CONTWARN=YES─┐
►►───┼──────────────┼──────────────────────────────────────────────────►◄
     └─CONTWARN=NO──┘
```

YES
the assembler issues diagnostic messages ASMA430W through ASMA433W when one of the following situations occurs:
- The operand on the continued record ends with a comma and a continuation statement is present but continuation does not start in the continue column (normally column 16).
- A list of one or more operands ends with a comma, but the continuation column (normally column 72) is blank.
- The continuation record starts in the continue column (normally column 16) but there is no comma present following the operands on the previous record.
- The continued record is full but the continuation record does not start in the continue column (normally column 16).

NO the assembler does not issue diagnostic messages ASMA430W through ASMA433W when an inconsistent continuation is encountered.

Default
CONTWARN=YES

Note: This option can be specified as an assembler invocation parameter by specifying the FLAG(CONT) option.

DBCS

```
      ┌─DBCS=NO──┐
►►────┼──────────┼──────────────────────────────────────────────────────►◄
      └─DBCS=YES─┘
```

YES
instructs the assembler to accept double-byte character set data, and to support graphic (G-type) constants. The assembler recognizes X'0E' and X'0F' in character strings enclosed by apostrophes, and treats them as shift-out and shift-in control characters for delimiting DBCS data.

NO the assembler does not recognize X'0E' and X'0F' as double-byte character set data delimiters, and does not support graphic (G-type) constants.

Default
DBCS=NO

DECK

```
      ┌─DECK=NO──┐
►►────┼──────────┼──────────────────────────────────────────────────────►◄
      └─DECK=YES─┘
```

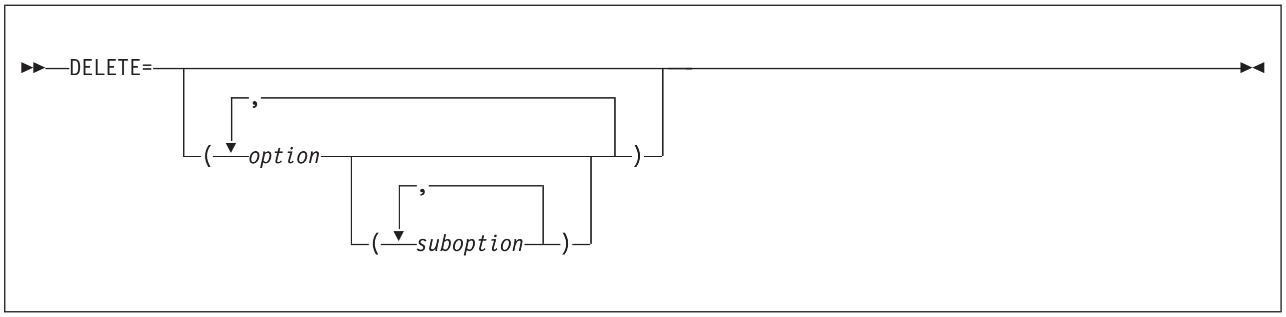
YES
instructs the assembler to place the generated object module in the file defined by the SYSPUNCH DD statement (z/OS), the FILEDEF SYSPUNCH command (CMS), or the ASSGN SYSPCH statement (z/VSE).

NO instructs the assembler not to place the generated object module in the file defined by the SYSPUNCH DD statement (z/OS), the FILEDEF SYSPUNCH command (CMS), or the ASSGN SYSPCH statement (z/VSE).

Default
DECK=NO

Important:
Under z/VSE, the DECK assembler option can only be set using the `// OPTION DECK` job control statement. The value specified for the DECK option during installation is ignored by the assembler.

DELETE



option
an option that is to have its installation default value fixed. This option cannot be overridden by invocation parameters. The option can be any of the following assembler options:

Options

ADATA	GOFF	OBJEXIT	SUBSTRWARN
ADEXIT	GOFFADATA	OPTABLE	SUPRWARN
ALIGN	IMPLENWARN	OPTABLELIST	SYSPARMV
ALIGNWARN	INEXIT	PAGE0WARN	TERM
ASA	INFO	PCONTROL[3]	TEST
BATCH	LANGUAGE	PROFILE	THREAD
CODEPAGE	LIBEXIT	PRTEXIT	TRANSLATE
COMPAT[1]	LIBMAC	PUSHWARN	TRMEXIT
CONTWARN	LIMIT	RA2	TYPECHECK[4]
DBCS	LINECOUNT	RECORDINFO	USING0WARN
DECK	LIST[2]	RENT	WARN
DXREF	MACHINE	RLD	WORKFILE
ESD	MACHINELIST	RXREF	XOBJADATA
EXLITW	MAP	SECTALGN	XOBJECT
FLAG	MXREF	SIZE	XREF
FOLD	OBJECT	STORAGE	

suboption
a suboption of certain options that is to have its installation default value fixed. This suboption cannot be overridden by invocation parameters.

Notes:

1. If you specify the COMPAT option, one or more of the following suboptions must also be specified. You cannot specify any other suboption of COMPAT.
   - CASE
   - LITTYPE
   - MACROCASE
   - SYSLIST
2. (z/OS and CMS) If you specify the LIST option, one of more of the following suboptions must be specified:
   - 121
   - 133
   - MAX
3. If you specify the PCONTROL option, one or more of the following suboptions must also be specified.

DATA	NODATA
GEN	NOGEN
MCALL	NOMCALL
MSOURCE	NOMSOURCE
ON	OFF
UHEAD	NOUHEAD

4. If you specify the TYPECHECK option, one of more of the following suboptions must be specified. You cannot specify any other suboption of TYPECHECK.
   - MAGNITUDE
   - REGISTER

Default
No options deleted.

DSECT

```
      ┌─DSECT=NO──┐
►►────┼───────────┼──────────────────────────────────────────────────►◄
      └─DSECT=YES─┘
```

YES
instructs the assembler to produce a DSECT called ASMAOPT.

NO instructs the assembler to produce a CSECT called ASMADOPT.

Default
DSECT=NO

DXREF

```
      ┌─DXREF=YES─┐
►►────┼───────────┼──────────────────────────────────────────────────►◄
      └─DXREF=NO──┘
```

YES
the *DSECT Cross Reference* section is generated as part of the assembler listing. The DSECT cross reference includes the symbolic names of all DSECTs defined in the assembly, the assembled length and the external symbol dictionary identification number (ESDID) of each DSECT, and the number of the statement where the definition of the DSECT began.

NO the *DSECT Cross Reference* section is not generated.

Default
DXREF=YES

ESD

```
      ┌─ESD=YES─┐
►►────┤         ├───────────────────────────────────────────────────────►◄
      └─ESD=NO──┘
```

YES
the *External Symbol Dictionary* (ESD) section is generated as part of the assembler listing. The ESD section of the assembler listing contains the external symbol dictionary information that is passed to the linkage editor or loader in the object module.

NO the ESD section is not included in the assembler listing.

Default
ESD=YES

EXLITW

```
      ┌─EXLITW=YES─┐
►►────┤            ├────────────────────────────────────────────────────►◄
      └─EXLITW=NO──┘
```

YES
the assembler issues the warning diagnostic message ASMA016W when a literal is the object of an EX instruction

NO the assembler suppresses the warning diagnostic message ASMA016W when a literal is specified as the object of an EX instruction.

Default
EXLITW=YES

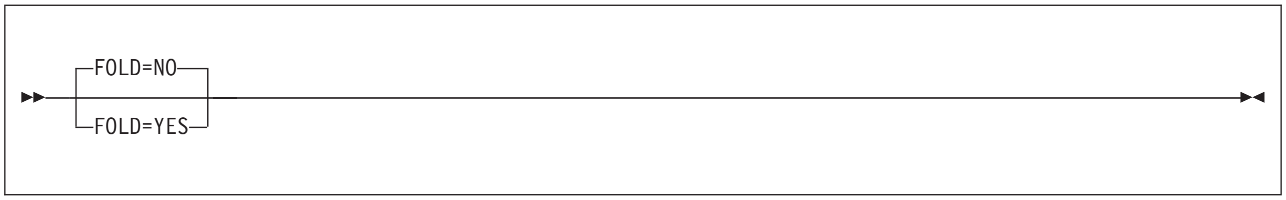
FLAG

```
      ┌─FLAG=0───────┐
►►────┤              ├──────────────────────────────────────────────────►◄
      └─FLAG=integer─┘
```

integer
error diagnostic messages with a severity code of *integer* or higher, appear in the assembler listing. Error diagnostic messages with a severity code lower than *integer* do not appear in the listing, and the severity codes associated with those messages are not used to set the return code issued by the assembler. Any severity code from 0 through 255 can be specified. Error diagnostic messages have a severity code of 0, 2, 4, 8, 12, 16, or 20. MNOTEs can have a severity code of 0 through 255.

Default
FLAG=0

FOLD

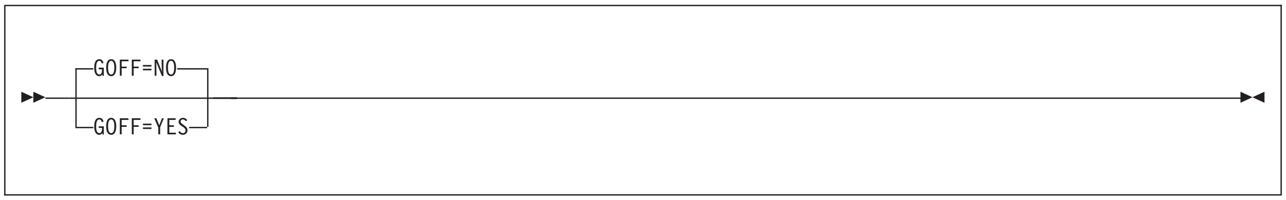


YES
: instructs the assembler to translate lowercase alphabetic characters (*a* through *z*) in the assembler listing to uppercase alphabetic characters (*A* through *Z*). All lowercase alphabetic characters are translated, including lowercase characters in source statements, assembler error diagnostic messages, and assembler listing lines provided by a user exit. Lowercase alphabetic characters are translated to uppercase alphabetic characters, regardless of the setting of the COMPAT(CASE) option.

NO lowercase alphabetic characters are not translated to uppercase alphabetic characters.

Default
: FOLD=NO

GOFF (z/OS and CMS)



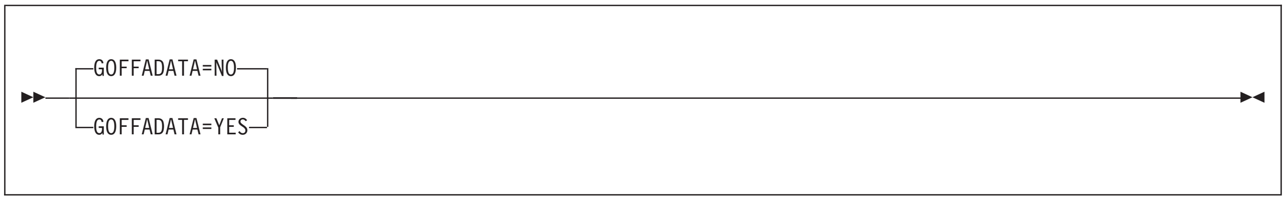
YES
: the assembler produces a generalized object format data set. The object data set is defined by the SYSLIN DD statement (z/OS) or the FILEDEF SYSLIN command (CMS). The generalized object format data set can only be processed by DFSMS/MVS™ 1.3 or later.

NO the assembler does not produce a generalized object format data set.

Default
: GOFF=NO

Notes:
1. The option XOBJECT is treated as a synonym for the GOFF option.
2. If you specify GOFF=YES, then you cannot specify TEST=YES.
3. If you specify GOFF=YES, you must also specify LIST=133 or LIST=MAX.

GOFFADATA (z/OS and CMS)



YES
the assembler includes the ADATA text record in the extended object format data set (if one is produced — see GOFF). This format requires GOFF=YES.

NO the assembler does not include the ADATA text record.

Default
GOFFADATA=NO

Notes:

1. This option can be specified as an assembler invocation parameter by specifying the GOFF(ADATA) option.
2. The option XOBJADATA is treated as a synonym for the GOFFADATA option.

IMPLENWARN

```
   ┌─IMPLENWARN=NO──┐
►►─┼────────────────┼──────────────────────────────────────────────────────►◄
   └─IMPLENWARN=YES─┘
```

YES
instructs the assembler to issue diagnostic message `ASMA169I` when an explicit length subfield is omitted from an SS-format machine instruction.

NO instructs the assembler not to issue diagnostic message `ASMA169I` when an explicit length subfield is omitted from an SS-format machine instruction.

Default
IMPLENWARN=NO

Note: This option can be specified as an assembler invocation parameter by specifying the FLAG(IMPLEN) option.

INEXIT

```
►►──INEXIT=──┬──────────────────────────┬──────────────────────────────────►◄
             └─(name─┬──────────┬─)─┘
                     └─,string──┘
```

name
identifies the name of a module that is loaded and called by the assembler to obtain source program statements, or to monitor the source program statements read by the assembler from SYSIN (z/OS and CMS), or from SYSIPT (z/VSE).

string
character string that is passed to the exit module as part of the parameter list built by the assembler. The character string is up to 64 characters in length. Any character can be included in the string, subject to the rules for building character strings defined in *HLASM Language Reference*. If the string includes blanks, commas, or parentheses, it must be enclosed in apostrophes.

Default
No exit specified.

Note: This option can be specified as an assembler invocation parameter by specifying the EXIT(INEXIT(mod1(str1))) option. For more information, see the *HLASM Programmer's Guide*.

INFO

```
     ┌─INFO=NO────────────┐
►►───┤                    ├──────────────────────────────────────────────────────►◄
     └─INFO=──┬─yyyymmdd─┬┘
              └─YES──────┘
```

NO Instructs the assembler not to copy any product information to the list dataset.

yyyymmdd
Instructs the assembler not to copy to the list dataset any product information which is dated prior to *yyyymmdd*.

YES
Instructs the assembler to copy all product information to the list dataset.

Default
INFO=NO

LANGUAGE

```
     ┌─LANGUAGE=EN──────┐
►►───┤                  ├────────────────────────────────────────────────────────►◄
     └─LANGUAGE=──┬─DE─┬┘
                  ├─ES─┤
                  ├─JP─┤
                  └─UE─┘
```

EN diagnostic messages issued by the assembler and assembler listing headings are printed in mixed uppercase and lowercase English.

DE diagnostic messages issued by the assembler are in German. Assembler listing headings are printed in mixed case English.

ES diagnostic messages issued by the assembler are in Spanish. Assembler listing headings are printed in mixed case English.

JP diagnostic messages issued by the assembler are in Japanese. Assembler listing headings are printed in uppercase English.

UE diagnostic messages issued by the assembler and assembler listing headings are printed in uppercase English.

Default
LANGUAGE=EN

LIBEXIT

```
►►──LIBEXIT=──┬────────────────────────┬──────────────────────────────────────►◄
              └─(name─┬─────────┬─)─┘
                      └─,string─┘
```

name
: identifies the name of a module that is loaded and called by the assembler to obtain macro library or copy library statements, or to monitor the macro library or copy library statements read by the assembler from the file defined by the SYSLIB DD statement (z/OS), the GLOBAL MACLIB command (CMS), or the LIBDEF SOURCE statement (z/VSE). Macro library statements are macro definition statements contained in a macro library. Copy library statements are source program statements contained in a copy library.

string
: the character string that is passed to the exit module as part of the parameter list built by the assembler. The character string is up to 64 characters in length. Any character can be included in the string, subject to the rules for building character strings defined in *HLASM Language Reference*. If the string includes blanks, commas, or parentheses, it must be enclosed in apostrophes.

Default
: No exit specified.

Note: This option can be specified as an assembler invocation parameter by specifying the EXIT(LIBEXIT(mod2(str2))) option. For more information, see the *HLASM Language Reference*.

LIBMAC

```
      ┌─LIBMAC=NO──┐
►►────┴─LIBMAC=YES─┴─────────────────────────────────────────────────────────►◄
```

YES
: macro definition statements read from a macro library are embedded in the input source program immediately preceding the first invocation of that macro. The assembler assigns statement numbers to the macro definition statements as though they were included in the input source program.

NO macro definition statements read from a macro library are not included in the input source program.

Default
: LIBMAC=NO

LIMIT

```
     ┌─LIMIT=NO──────┐
►►───┼───────────────┼─────────────────────────────────────────────►◄
     └─LIMIT=integer─┘
```

integer
: when specified with the WARN=8 suboption of the USING option, tells the assembler the maximum displacement that is allowed in base-displacement address resolution before a warning message is issued. When the assembler converts an implicit address (symbolic address) into an explicit address (base-displacement form address), it checks the calculated displacement. If the calculated displacement is greater than the value specified by *integer*, message ASMA304 is issued. *integer* must be a decimal value in the range 0 to 4095. Specifying a value of 4095 is equivalent to specifying LIMIT=NO.

NO no calculated displacement checking is done.

Default
: LIMIT=NO

Note: This option can be specified as an assembler invocation parameter by specifying the LIMIT suboption of the USING option.

LINECOUNT

```
     ┌─LINECOUNT=60──────┐
►►───┼───────────────────┼─────────────────────────────────────────►◄
     └─LINECOUNT=integer─┘
```

integer
: the number of lines printed on each page of the assembler listing. *integer* must have a value of 0, or 10 to 3267. If a value of 0 is specified, no page ejects are generated and EJECT, CEJECT, and TITLE statements in the assembly are ignored.

  Up to 9 lines on each page can be used for heading lines.

Default
: LINECOUNT=60

LIST

```
                        (1)
     ┌─LIST=─┬─121─┬──────┐
     │       └─YES─┘      │
►►───┤                    ├────────────────────────────────────────────────────►◄
     └─LIST──=──┬─133─┬───┘
                ├─MAX─┤
                └─NO──┘
```

Notes:

1 For z/OS and CMS, the default is 121, for z/VSE the default is YES.

121 (z/OS and CMS only)
The *Source and Object* section of the assembler listing is produced in the 121-character wide format.

133 (z/OS and CMS Only)
The *Source and Object* section of the assembler listing is produced in the 133-character wide format.

MAX (z/OS and CMS only)
The *Source and Object* section of the assembler listing is produced in:

121-character wide format
if the logical record length (LRECL) of the listing data set is less than 133.

133-character wide format
if the LRECL of the listing data set is 133 or more.

YES
same as 121.

NO no assembler listing is produced. If LIST=NO is specified, the options DXREF, ESD, MAP, MXREF, PCONTROL, PRTEXIT, RLD, and XREF are ignored.

Defaults

z/OS and CMS
LIST=121

z/VSE LIST=YES

Note: (z/OS and CMS only) If XOBJECT=YES is specified, LIST=133 or LIST=MAX must be specified.

MACHINE

```
►►──MACHINE──=──┬─────────────┬──────────────────────────────────────────────►◄
                ├─S370────────┤
                ├─S370XA──────┤
                ├─S370ESA─────┤
                ├─S390────────┤
                ├─S390E───────┤
                ├─ZSERIES─────┤
                ├─ZS──────────┤
                ├─ZSERIES-2───┤
                ├─ZS-2────────┤
                ├─ZSERIES-3───┤
                ├─ZS-3────────┤
                ├─ZSERIES-4───┤
                ├─ZS-4────────┤
                ├─ZSERIES-5───┤
                ├─ZS-5────────┤
                ├─ZSERIES-6───┤
                └─ZS-6────────┘
```

Usage: The MACHINE option is a synonym of the OPTABLE option, and it's operands are also synonyms of, but are not identical to, the OPTABLE operands. See “OPTABLE” on page 161.

S370
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to System/370™ systems, including those with a vector facility. See OPTABLE=370.

S370XA
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to systems operating in System/370 extended-architecture mode, including those with a vector facility. See OPTABLE=XA.

S370ESA, S390, S390E
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to systems operating according to the ESA/370 or ESA/390 architecture, including those with a vector facility. See OPTABLE=ESA.

ZSERIES, ZS
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems. See OPTABLE=ZOP.

ZSERIES-2, ZS-2
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the Long-Displacement Facility, the DAT Enhancement Facility, the Message-Security Assist, and the HFP Multiply-Add/Subtract Facility. See OPTABLE=YOP.

ZSERIES-3, ZS-3
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the decimal floating point facility. See OPTABLE=ZS3.

ZSERIES-4, ZS-4
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the general instructions extensions facility. See OPTABLE=ZS4.

ZSERIES-5, ZS-5
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the general instructions extensions facility and z196 instructions. See OPTABLE=ZS5.

ZSERIES-6, ZS-6
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the general instructions extensions facility and zEnterprise EC12 (zEC12) instructions. See OPTABLE=ZS6.

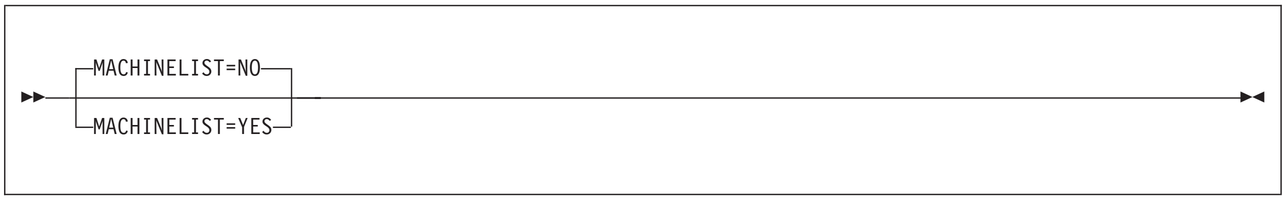
Default
: There is no default for the MACHINE option.

If you do not specify any value for the MACHINE option, the assembler takes the value you have specified for OPTABLE, if any, or the default OPTABLE=UNI, if not.

If you specify a value for MACHINE and a value for OPTABLE, and the values you specify are mutually exclusive (for example, you specify MACHINE=S390 and OPTABLE=ZOP) then the assembler will take the specification for MACHINE.

Note: There is no MACHINE option equivalent to the OPTABLE=DOS option.

MACHINELIST



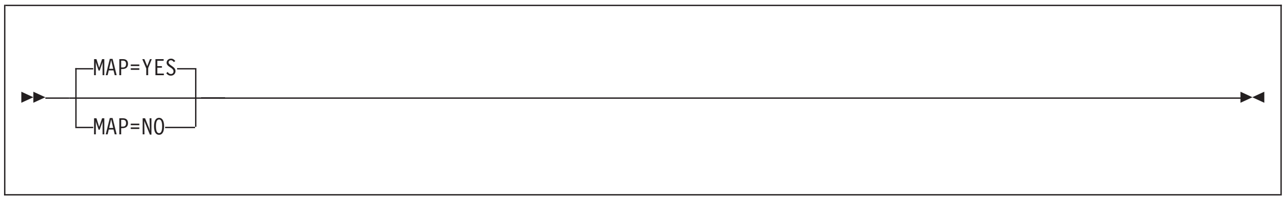
Note: The MACHINELIST option is synonymous with the OPTABLELIST option.

NO instructs the assembler not to produce the Operation Code Table Contents report.

YES
: instructs the assembler to produce the Operation Code Table Contents report.

Default
: MACHINELIST=NO

MAP



YES
: the *USING Map* section is generated as part of the assembler listing. The USING Map is a summary

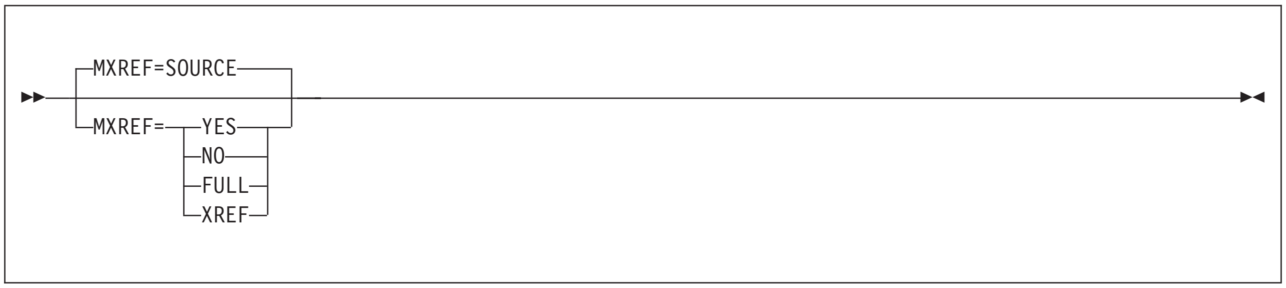
of each USING, DROP, PUSH USING, and POP USING statement in the assembler, including the statement number, the text of the statement, and the registers involved.

NO the USING map is not generated.

Default
MAP=YES

Note: This option can be specified as an assembler invocation parameter by specifying the MAP suboption of the USING option.

MXREF



SOURCE
the *Macro and Copy Code Source Summary* section only is generated as part of the assembler listing. The *Macro and Copy Code Cross Reference* section is not.

YES
same as SOURCE.

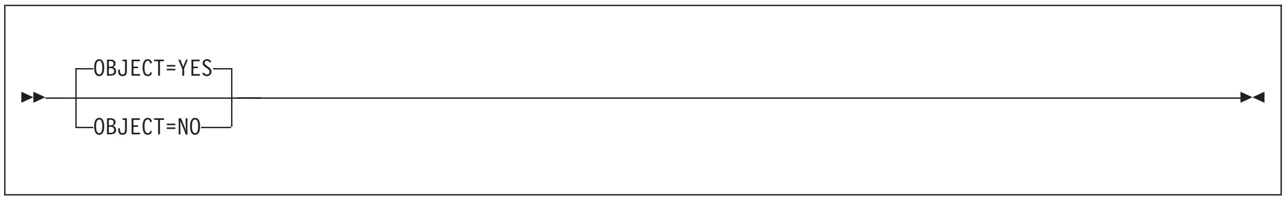
NO neither the *Macro and Copy Code Source Summary* section nor the *Macro and Copy Code Cross Reference* section are generated as part of the assembler listing.

FULL
the *Macro and Copy Code Source Summary* and *Macro and Copy Code Cross Reference* sections are generated as part of the assembler listing. These sections include the name of each macro library or copy library accessed, the volume serial number of the first DASD volume on which the library resides, and the names of each member retrieved from the library.

XREF
the *Macro and Copy Code Cross Reference* section only is generated as part of the assembler listing. The *Macro and Copy Code Source Summary* is not.

Default
MXREF=SOURCE

OBJECT



YES
instructs the assembler to place the generated object module in the file defined by the SYSLIN DD statement (z/OS), the FILEDEF SYSLIN command (CMS), or the DLBL IJSYSLN statement (z/VSE).

NO instructs the assembler not to place the generated object module in the file defined by the SYSLIN DD statement (z/OS), the FILEDEF SYSLIN command (CMS), or the DLBL IJSYSLN statement (z/VSE).

Default
OBJECT=YES

OBJEXIT

```
►►──OBJEXIT=──┬──────────────────────────┬──────────────────────────────►◄
              └─(name─┬──────────┬─)─┘
                      └─,string──┘
```

name
identifies the name of a module that is loaded and called by the assembler to receive object module records, or to monitor the object module records written by the assembler to SYSPUNCH or SYSLIN (z/OS and CMS), or SYSPCH (z/VSE).

string
the character string that is passed to the exit module as part of the parameter list built by the assembler. The character string is up to 64 characters in length. Any character can be included in the string, subject to the rules for building character strings defined in *HLASM Language Reference*. If the string includes blanks, commas, or parentheses, it must be enclosed in apostrophes.

Default
No exit specified.

Note: This option can be specified as an assembler invocation parameter by specifying the EXIT(OBJEXIT(mod4(str4))) option. For more information, see the *HLASM Programmer's Guide*.

OPTABLE

```
   ┌─OPTABLE=UNI──────┐
►►─┴─OPTABLE=──┬─DOS─┬┴──────────────────────────────────────────────────►◄
               ├─ESA─┤
               ├─XA──┤
               ├─370─┤
               ├─ZOP─┤
               ├─YOP─┤
               ├─ZS3─┤
               ├─ZS4─┤
               ├─ZS5─┤
               └─ZS6─┘
```

UNI
the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions that can be used on all System/370 and System/390® systems, including those with a vector facility.

DOS
: the assembler loads and uses the DOS operation code table. The DOS operation code is designed specifically for assembling programs previously assembled using the DOS/VSE assembler. The operation code table contains the System/370 machine instructions, excluding those with a vector facility.

ESA
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to systems operating according to the ESA/370 or ESA/390 architecture, including those with a vector facility.

XA
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to systems operating in System/370 extended-architecture mode, including those with a vector facility.

370
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to System/370 systems, including those with a vector facility.

ZOP
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems.

YOP
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the Long-Displacement Facility, the DAT Enhancement Facility, the Message-Security Assist, and the HFP Multiply-Add/Subtract Facility.

ZS3
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the decimal floating point facility.

ZS4
: the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the general instructions extensions facility.

| **ZS5**
| : the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the general instructions extensions facility and z196 instructions.

| **ZS6**
| : the assembler loads and uses the operation code table that contains the symbolic operation codes for the machine instructions specific to Z/Architecture systems with the general instructions extensions facility and zEnterprise EC12 (zEC12) instructions.

Default
: OPTABLE=UNI

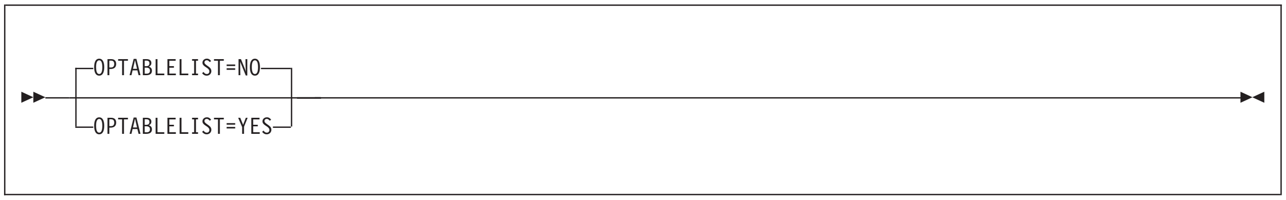
Usage:

1. The OPTABLE option is synonymous with the MACHINE option. If you specify a value for OPTABLE that conflicts with the value you have specified for MACHINE, the specification for MACHINE takes precedence. See "MACHINE" on page 158 for more information.
2. These operation code tables do not contain symbolic operation codes for machine instructions that are unique to IBM 4300 Processors operating in ECPS:VSE mode.
3. The operation codes supported by High Level Assembler are described in:

   Enterprise Systems Architecture/390 Principles of Operation, SA22-7201

   z/Architecture Principles of Operation, SA22-7832 *z/Architecture Reference Summary*, SA22-7871

OPTABLELIST

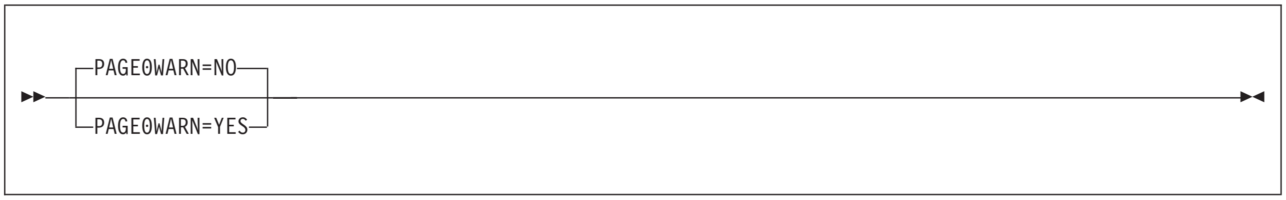


NO instructs the assembler not to produce the Operation Code Table Contents report.

YES
instructs the assembler to produce the Operation Code Table Contents report.

Default
OPTABLELIST=NO

PAGE0WARN



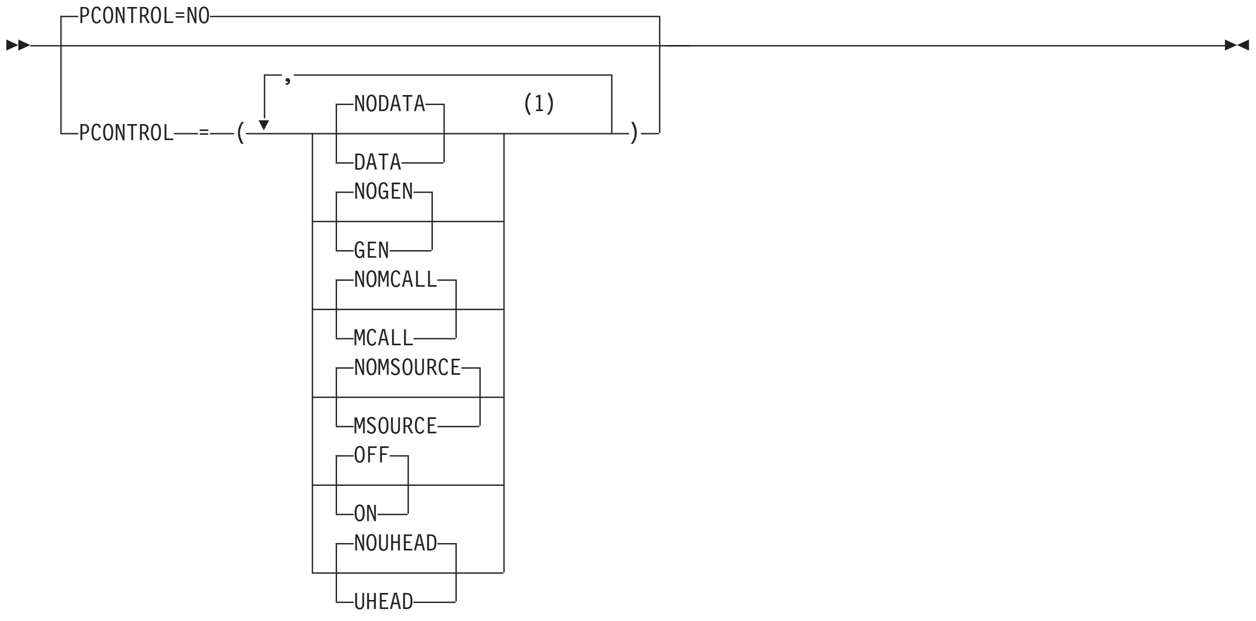
NO instructs the assembler not to issue diagnostic message ASMA309W when an operand is resolved to a baseless address and a base and displacement is expected.

YES
instructs the assembler to issue diagnostic message ASMA309W when an operand is resolved to a baseless address and a base and displacement is expected. Since a baseless address is generally correct in the case of load address and all shift instructions, this message is not issued for any of these instructions.

Default
PAGE0WARN=NO

Note: This option can be specified as an assembler invocation parameter by specifying the FLAG(PAGE0) option.

PCONTROL



Notes:

1 Choose at least one option.

DATA
instructs the assembler to print the object code of all constants in full, as though a PRINT DATA statement was specified at the beginning of the source program. All PRINT NODATA statements in the source program are ignored. However, specifying PCONTROL=DATA does not override PRINT OFF or PRINT NOGEN statements in the source program.

NODATA
instructs the assembler to print only the first eight bytes of the object code of constants, as though a PRINT NODATA statement was specified at the beginning of the source program. All PRINT DATA statements in the source program are ignored.

GEN
instructs the assembler to print all statements generated by the processing of a macro, as though a PRINT GEN statement was specified at the beginning of the source program. All PRINT NOGEN statements in the source program are ignored. However, specifying PCONTROL=GEN does not override PRINT OFF statements in the source program.

NOGEN
instructs the assembler not to print statements generated by the processing of a macro or open code statements with substitution variables, as though a PRINT NOGEN statement was specified at the beginning of the source program. All PRINT GEN and PRINT MSOURCE statements in the source program are ignored.

MCALL
instructs the assembler to print nested macro instructions, as though a PRINT MCALL statement was specified at the beginning of the source program. All PRINT NOMCALL statements in the source program are ignored. However, specifying PCONTROL=MCALL does not override PRINT OFF or PRINT NOGEN statements in the source program.

NOMCALL
instructs the assembler not to print nested macro instructions, as though a PRINT NOMCALL statement was specified at the beginning of the source program. All PRINT MCALL statements in the source program are ignored.

MSOURCE
instructs the assembler to print the source statements generated during macro processing and the assembled addresses and generated object code of the statements. All PRINT NOMSOURCE statements in the source program are ignored. However, specifying PCONTROL=MSOURCE does not override PRINT OFF or PRINT NOGEN statements in the source program.

NOMSOURCE
instructs the assembler not to print source statements generated during macro processing, but print the assembled addresses and generated object code of the statements. All PRINT MSOURCE statements in the source program are ignored.

ON instructs the assembler to produce an assembler listing unless the LIST=NO option is specified. All PRINT OFF statements in the source program are ignored.

OFF
instructs the assembler not to produce the *Source and Object* section of the assembler listing. All PRINT ON statements in the source program are ignored.

UHEAD
instructs the assembler to print a summary of active USINGs in the heading lines of each page of the *Source and Object* section of the listing, as though a PRINT UHEAD statement was specified at the beginning of the source program. All PRINT NOUHEAD statements in the source program are ignored. However, specifying PCONTROL=UHEAD does not override PRINT OFF statements in the source program.

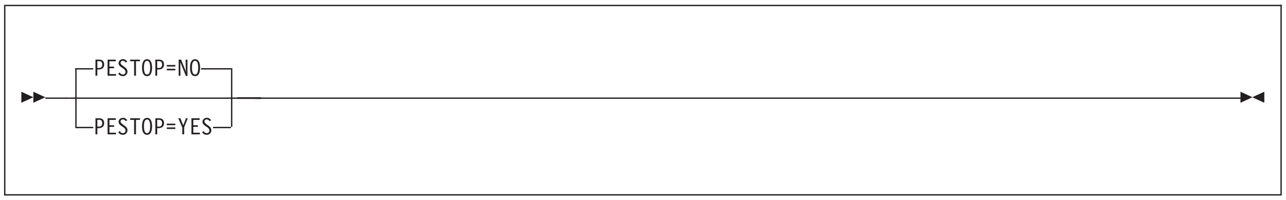
NOUHEAD
instructs the assembler not to print a summary of active USINGs, as though a PRINT NOUHEAD statement was specified at the beginning of the source program. All PRINT UHEAD statements in the source program are ignored.

NO the assembler honors all PRINT statements in the source program. The standard PRINT operands active at the beginning of an assembly are ON, GEN, NODATA, MSOURCE, and UHEAD.

Default
PCONTROL=NO

Note: The PCONTROL option cannot be used to override the LIST=NO option. If the LIST=NO option is specified, the PCONTROL option is ignored.

PESTOP



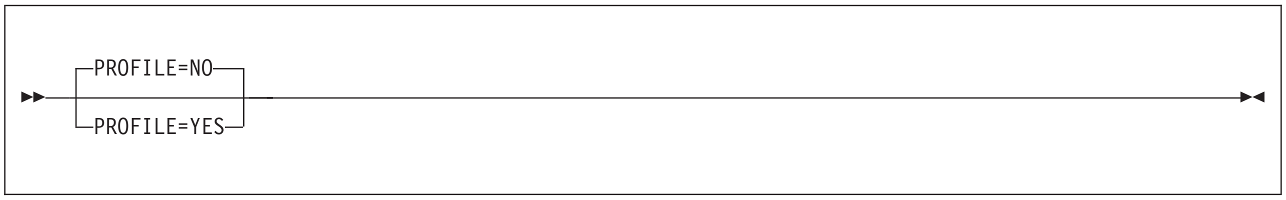
YES
if the assembler detects errors in the invocation parameters or in any *PROCESS statements specified when the assembler is called, all such errors are reported in assembler error diagnostic messages, and the assembly terminates. Similarly, if the invocation parameters or *PROCESS statements contain any options that have a fixed installation default value (the option was specified in the DELETE option of

the ASMAOPT macro), all such attempts to override a fixed installation default are reported in assembler error diagnostic messages, and the assembly terminated. See the DELETE option onpage "DELETE" on page 149.

NO the assembler reports errors in invocation parameters and in any *PROCESS statements, and continues the assembly using the installation default value for the erroneously specified option. Similarly, the assembler reports any attempts to override fixed installation defaults, and continues the assembly using the default value.

Default
PESTOP=NO

PROFILE



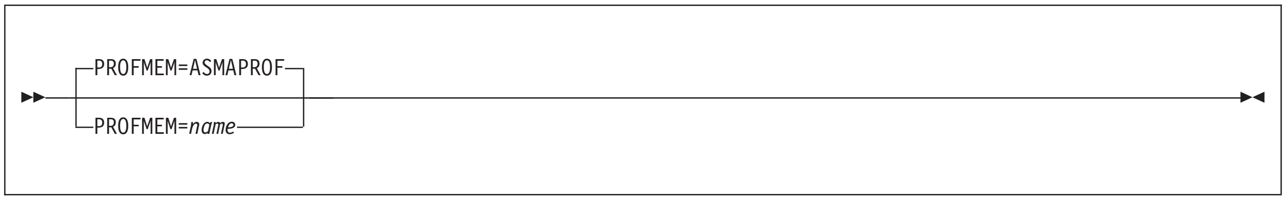
YES
the assembler copies the profile member into the source program, as if the source program contained a COPY instruction as its first statement.

If you do not specify a profile member using the PROFMEM operand, the default member name is ASMAPROF.

NO the assembler does not copy the profile member into the source program.

Default
PROFILE=NO

PROFMEM



ASMAPROF
the default profile member.

name
the profile member name.

Default
PROFMEM=ASMAPROF

See also "PROFILE."

Note: The member name can be specified as an assembler invocation parameter by specifying the PROFILE(name) option.

PRTEXIT

```
►►──PRTEXIT=──┬──────────────────────┬──────────────────────────────────────►◄
              └─(name─┬─────────┬─)─┘
                      └─,string─┘
```

name
identifies the name of a module that is loaded and called by the assembler to receive assembler listing records, or to monitor the assembler listing records written by the assembler to SYSPRINT (z/OS and CMS) or SYSLST (z/VSE).

string
the character string that is passed to the exit module as part of the parameter list built by the assembler. The character string is up to 64 characters in length. Any character can be included in the string, subject to the rules for building character strings defined in *HLASM Language Reference*. If the string includes blanks, commas, or parentheses, it must be enclosed in apostrophes.

Default
No exit specified.

Note: This option can be specified as an assembler invocation parameter by specifying the EXIT(PRTEXIT(mod3(str3))) option. For more information, see the *HLASM Programmer's Guide*.

PUSHWARN

```
   ┌─PUSHWARN=YES─┐
►►─┴─PUSHWARN=NO──┴──────────────────────────────────────────────────────────►◄
```

YES
the assembler issues warning diagnostic message ASMA138W when a PUSH/POP stack is not empty at the completion of an assembly.

NO the assembler suppresses warning diagnostic message ASMA138W when a PUSH/POP stack is not empty at the completion of an assembly.

Default
PUSHWARN=YES

RA2

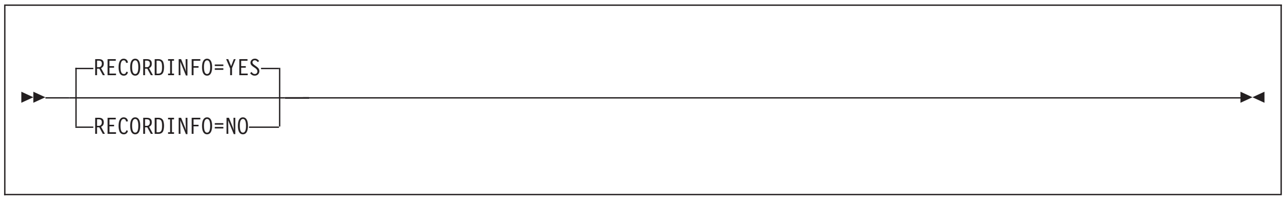
```
   ┌─RA2=NO──┐
►►─┴─RA2=YES─┴───────────────────────────────────────────────────────────────►◄
```

YES
the assembler suppresses error diagnostic message ASMA066W when 2-byte relocatable address constants, such as AL2(*) and Y(*), are defined in the source program.

NO the assembler issues error diagnostic message ASMA066W when 2-byte relocatable address constants, such as AL2(*) and Y(*), are defined in the source program.

Default
RA2=NO

RECORDINFO



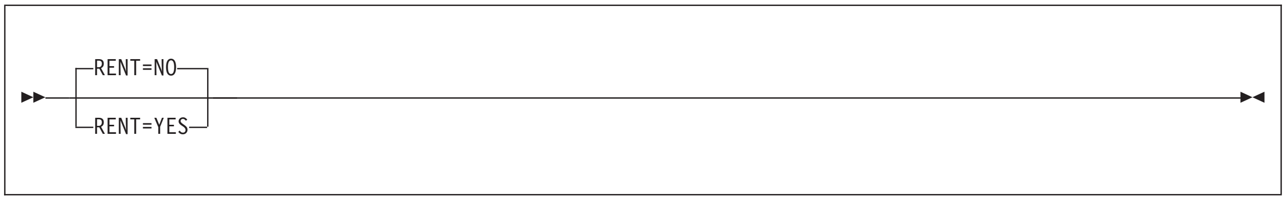
YES
instructs the assembler to do the following:
- Issue diagnostic message ASMA435I immediately after the last diagnostic message for each statement in error. The message text describes the record number and input data set name of the statement in error.
- Include the member name (if applicable), the record number and the input data set concatenation value with the statement number in the list of flagged statements in the *Diagnostic Cross Reference and Assembler Summary* section of the assembler listing.

NO instructs the assembler to do the following:
- Not issue diagnostic message ASMA435I for statements in error.
- Only show the statement number in the list of flagged statements in the *Diagnostic Cross Reference and Assembler Summary* section of the assembler listing.

Default
RECORDINFO=YES

Note: This option can be specified as an assembler invocation parameter by specifying the FLAG(RECORD) option.

RENT



YES
the assembler checks for possible coding violations of program reenterability.

NO the assembler does not check for possible coding violations of program reenterability except in executable control sections defined with the RSECT instruction.

Default
RENT=NO

RLD

```
     ┌─RLD=YES─┐
►►───┼─────────┼──────────────────────────────────────────────────────►◄
     └─RLD=NO──┘
```

YES
the *Relocation Dictionary* (RLD) section is generated as part of the assembler listing. The RLD section of the assembler listing contains the relocation list dictionary information that is passed to the linkage editor or loader in the object module.

NO the RLD is not included in the assembly listing.

Default
RLD=YES

RXREF

```
     ┌─RXREF=YES─┐
►►───┼───────────┼────────────────────────────────────────────────────►◄
     └─RXREF=NO──┘
```

YES
Instructs the assembler to produce the *General Purpose Register Cross Reference* section of the assembler listing. The General Purpose Register Cross Reference includes:
- The register number
- The statement number which references the register

NO Instructs the assembler not to produce the *General Purpose Register Cross Reference* section of the assembler listing.

Default
RXREF=YES

SECTALGN

```
     ┌─SECTALGN=8──────────┐
►►───┼─────────────────────┼──────────────────────────────────────────►◄
     └─SECTALGN=alignment──┘
```

alignment
Specifies the desired alignment for all sections, expressed as a power of 2 with a range from 8 (doubleword) to 4096 (page). The default value is 8 for doubleword alignment.

Default
SECTALGN=8

SIZE

```
     ┌─SIZE=MAX──────────────┐
►►───┤                       ├──────────────────────────────────────────────────►◄
     └─SIZE=─┬─intgK─────┬───┘
             ├─intgM─────┤
             ├─MAX-intgK─┤
             └─MAX-intgM─┘
```

SIZE=MAX
specifies that the assembler requests all the available space in the user region (z/OS), virtual machine (CMS), or in the partition GETVIS (z/VSE).

SIZE=*intg*K
specifies the amount of virtual storage in 1024-byte (1K) increments.

The minimum acceptable value is 200K.

SIZE=*intg*M
specifies the amount of virtual storage in 1048576-byte (1M) increments.

The minimum acceptable value is 1M.

SIZE=MAX-*intg*K
Specifies that the assembler requests all the available space in the user region (z/OS), virtual machine (CMS), or partition GETVIS (z/VSE), less the amount of *intg*K of storage (1K equals 1024 bytes).

The minimum acceptable value is 1K.

MAX-*intg*M
Specifies that the assembler requests all the available space in the user region (z/OS), virtual machine (CMS), or partition GETVIS (z/VSE), less the amount of *intg*M of storage (1M equals 1048756 bytes).

The minimum acceptable value is 1M.

Notes:
1. The maximum storage value you can specify might not be available in the user region (z/OS), virtual machine (CMS), or in the partition GETVIS (z/VSE),
2. The assembler obtains the storage specified from above the 16 MB line, if available. If storage above the 16 MB line is not available, the assembler obtains the storage from below the 16 MB line.
3. The minimum amount of working storage required by the assembler is 200K.
4. When you specify the MAX suboption, the assembler releases 128 K back to the user region (z/OS), virtual machine (CMS), or the partition GETVIS (z/VSE), for system usage. When you specify the MAX suboption, there might not be enough storage remaining in the user region (z/OS), virtual machine (CMS), or the partition GETVIS (z/VSE), to load any exits you specify, or any external functions you use in your assembly.
5. The assembler loads user I/O exits before it obtains the working storage. If the user exit obtains storage, then it reduces the amount available for the assembler.
6. The assembler loads external function routines after it obtains working storage. If you use external functions in your program, you should reduce the value you specify in the SIZE option, to allow storage space for the external function modules, and any storage they might acquire.

High Level Assembler acquires the amount of storage you specify in the SIZE option from the user region (z/OS), virtual machine (CMS) or partition GETVIS (z/VSE).

The statistics in the *Diagnostic Cross Reference and Assembler Summary* section of the assembly listing shows the amount of storage the assembler allocated. You must either specify a large enough value on the SIZE option to allow the assembler to perform an in-storage assembly, or use the WORKFILE option to provide a work data set. For information about the WORKFILE option, see page “WORKFILE” on page 176.

Default
SIZE=MAX

STORAGE

```
►►──STORAGE=──┬───────┬──────────────────────────────────────────────────────►◄
              ├─BELOW─┤
              └─ABOVE─┘
```

Note: Starting with High Level Assembler R5, the STORAGE option is obsolete. It is documented here for compatibility. The assembler ignores this option and always attempts to obtain the storage specified by the SIZE option from above the 16 MB line. If storage above the 16 MB line is not available, the assembler attempts to obtain the storage from below the 16 MB line.

SUBSTRWARN

```
     ┌─SUBSTRWARN=NO──┐
►►───┼────────────────┼──────────────────────────────────────────────────────►◄
     └─SUBSTRWARN=YES─┘
```

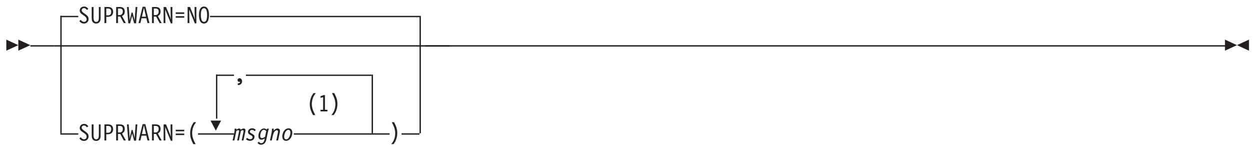
YES
the assembler issues error diagnostic message ASMA094 when the second subscript value of the substring notation indexes past the end of the character expression.

NO the assembler does not issue error diagnostic message ASMA094 when the second subscript value of the substring notation indexes past the end of the character expression.

Default
SUBSTRWARN=NO

Note: This option can be specified as an assembler invocation parameter by specifying the FLAG(SUBSTR) option.

SUPRWARN



Notes:

1 Specify at least one message number.

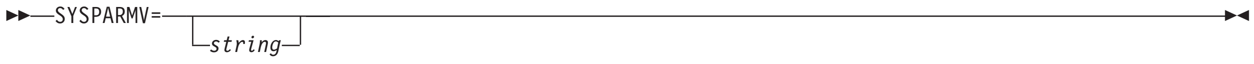
Usage: Use the SUPRWARN option to suppress specified message numbers, of warning (4) or less severity. Specify the 1-4 digit message number.

NO No messages will be suppressed.

msgno,msgno
The assembler will suppress the specified messages. Specify the 1-4 digit message number. For example, to suppress warning message ASMA066W, specify SUPRWARN=066.

Default
SUPRWARN=NO

SYSPARMV



string
the character string used as the default value of the &SYSPARM system variable symbol. The character string is up to 255 characters in length. Any character can be included in the string, subject to the rules for building character strings defined in *HLASM Language Reference*. If the string includes blanks, commas, or parentheses, it must be enclosed in apostrophes.

Default
SYSPARMV not specified.

Note: This option can be specified as an assembler invocation parameter by specifying the SYSPARM(string) option.

On z/VSE, if a SYSPARM value is provided in the default options, it is not possible to override it with a null string in the // OPTION statement. It is possible to override it only with a non-null value. unless you use the invocation parm string to do it.

TERM

```
     ┌─TERM=NO──────────────┐
►►───┤                      ├──────────────────────────────────────────────────►◄
     └─TERM=──┬─YES───────┬─┘
              ├─WIDE──────┤
              └─NARROW────┘
```

YES
equivalent to WIDE. See the description of TERM=WIDE below.

WIDE
assembler messages are written to the file defined by the SYSTERM DD statement (z/OS), the FILEDEF SYSTERM statement (CMS), or the ASSGN SYSLOG statement (z/VSE). Messages written to the terminal file do not have multiple consecutive blanks compressed to a single blank.

NARROW
assembler messages are written to the file defined by the SYSTERM DD statement (z/OS), the FILEDEF SYSTERM statement (CMS), or the ASSGN SYSLOG statement (z/VSE). Messages written to the terminal file have multiple consecutive blanks compressed to a single blank.

NO assembler messages are not written to the file defined by the SYSTERM DD statement (z/OS), the FILEDEF SYSTERM statement (CMS), or the ASSGN SYSLOG statement (z/VSE).

Default
TERM=NO

TEST

```
     ┌─TEST=NO───┐
►►───┤           ├─────────────────────────────────────────────────────────────►◄
     └─TEST=YES──┘
```

YES
the object module contains the special source symbol table (SYM records). TEST=YES cannot be specified with GOFF=YES or XOBJECT=YES.

NO the object module does not contain the special source symbol table (SYM records).

Default
TEST=NO

THREAD

```
     ┌─THREAD=YES─┐
►►───┤            ├────────────────────────────────────────────────────────────►◄
     └─THREAD=NO──┘
```

YES
instructs the assembler to not reset the location counter at the beginning of each CSECT.

NO instructs the assembler to reset the location counter at the beginning of each CSECT.

Default
THREAD=YES

TRANSLATE

```
    ┌─TRANSLATE=NO──────┐
►►──┼───────────────────┼──────────────────────────────────────────────────►◄
    └─TRANSLATE=──┬─AS─┬┘
                  └─xx─┘
```

AS use ASCII translation table ASMALTAS to translate characters contained in data constants and literals.

xx use user translation table ASMALT*xx* to translate characters contained in data constants and literals.

NO use the standard EBCDIC character set to translate characters contained in data constants and literals.

Default
TRANSLATE=NO

TRMEXIT

```
►►──TRMEXIT=──┬──────────────────────────┬─────────────────────────────────►◄
              └─(name──┬──────────┬──)───┘
                       └─,string──┘
```

name
the user-supplied terminal (SYSTERM) exit. This module is invoked for TERM exit type processing.

This exit might be used, for example, to write variable-length assembler terminal records.

string
the character string that is passed to the exit module as part of the parameter list built by the assembler. The character string is up to 64 characters in length. Any character can be included in the string, subject to the rules for building character strings defined in *HLASM Language Reference*. If the string includes blanks, commas, or parentheses, it must be enclosed in apostrophes.

Default
No exit specified.

Note: This option can be specified as an assembler invocation parameter by specifying the EXIT(TRMEXIT(mod6(str6))) option. For more information, see the *HLASM Programmer's Guide*.

TYPECHECK

```
   ┌─TYPECHECK=(MAGNITUDE,REGISTER)──────────┐
►►─┤                                         ├──────────────────────────────────────────────►◄
   └─TYPECHECK=─┬─(NOMAGNITUDE,REGISTER)───┬─┘
                ├─(MAGNITUDE,NOREGISTER)───┤
                ├─(NOMAGNITUDE,NOREGISTER)─┤
                ├─YES──────────────────────┤
                └─NO───────────────────────┘
```

MAGNITUDE | NOMAGNITUDE
Specifies whether or not the assembler will perform magnitude validation of signed immediate-data fields of machine instruction operands.

REGISTER | NOREGISTER
specifies whether or not the assembler will perform type checking of register fields of machine instruction operands.

YES
Enable all typechecking.

NO Disable all typechecking.

Default
TYPECHECK=(MAGNITUDE,REGISTER)

USING0WARN

```
   ┌─USING0WARN=YES─┐
►►─┤                ├──────────────────────────────────────────────────────────────────►◄
   └─USING0WARN=NO──┘
```

YES
the assembler issues warning diagnostic message ASMA306W when a USING:
- Is coincident with the implied USING 0,0 and option USING(WARN(1)) is in effect.
- Overlaps the implied USING 0,0 and option USING(WARN(4)) is in effect.

NO the assembler suppresses warning diagnostic message ASMA306W.

Default
USING0WARN=YES

WARN

```
   ┌─WARN=15──────────────┐
▶▶─┤                      ├──────────────────────────────────────────────────▶◀
   └─WARN=─┬─integer─┬────┘
           └─NO──────┘
```

integer
the conditions under which the assembler issues various warning messages pertaining to the USING instructions used in the source program. The permissible value for *integer* is in the range 1 to 15.

Several conditions can be combined by adding together the associated condition numbers. For example, specifying WARN(12) instructs the assembler to issue warning diagnostic messages for the conditions with condition numbers 4 and 8.

The meaning of the various condition number values for *integer* are:

1 the assembler issues message:
- ASMA300 when a earlier active ordinary (unlabeled) USING's range coincides with and supersedes that of the USING being processed.
- ASMA301 when the range of the USING being processed coincides with and supersedes that of a earlier active ordinary (unlabeled) USING.
- ASMA306 when the range of the USING being processed coincides with the implicit USING 0,0 (for example USING 0,2).

2 the assembler issues message ASMA302 when a USING specifies R0 as a base register, with a non-zero absolute or relocatable expression for the base address.

4 Multiple resolutions: The assembler issues message:
- ASMA303 when multiple resolutions are possible for an implicit address.
- ASMA306 when the range of the USING being processed overlaps the range of the implicit USING 0,0 (for example USING 16,2).

8 the assembler issues message ASMA304 when the calculated displacement in any valid resolution exceeds the threshold specified in the LIMIT suboption.

NO no USING warning messages are issued.

Default
WARN=15

Note: This option can be specified as an assembler invocation parameter by specifying the WARN suboption of the USING option.

WORKFILE

```
   ┌─WORKFILE=NO──┐
▶▶─┤              ├──────────────────────────────────────────────────────────▶◀
   └─WORKFILE=YES─┘
```

YES
the assembler will attempt to open the workfile dataset, and if successfully opened, then the workfile will be used as a spill file if required.

NO the assembler does not attempt to open the workfile dataset. Sufficient storage must be specified via the SIZE option to avoid abnormal terminations.

Default
WORKFILE=NO

XOBJADATA (z/OS and CMS)

```
      ┌─XOBJADATA=NO──┐
►►────┤               ├────────────────────────────────────────────►◄
      └─XOBJADATA=YES─┘
```

YES
the assembler includes the ADATA text record in the generalized object format data set (if one is produced — see XOBJECT). This format requires XOBJECT=YES.

NO the assembler does not include the ADATA text record.

Default
XOBJADATA=NO

Notes:
1. This option can be specified as an assembler invocation parameter by specifying the XOBJECT(ADATA) option.
2. This option has been superseded by the option GOFFADATA.

XOBJECT (z/OS and CMS)

```
      ┌─XOBJECT=NO──┐
►►────┤             ├──────────────────────────────────────────────►◄
      └─XOBJECT=YES─┘
```

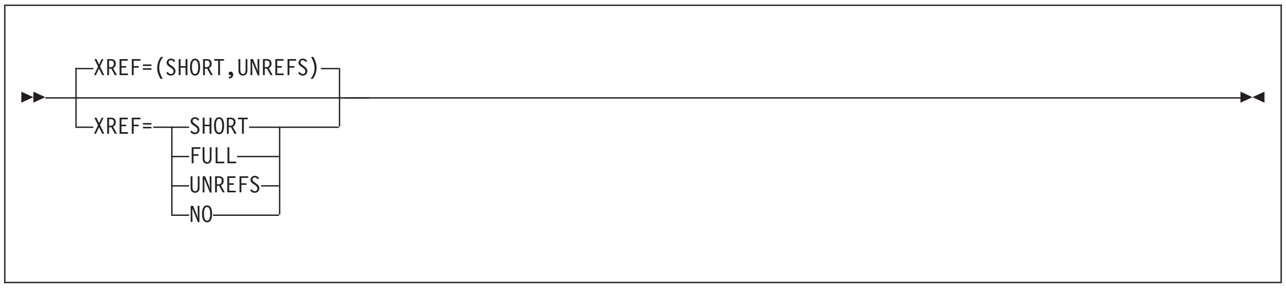
YES
the assembler produces a generalized object format data set. The object data set is defined by the SYSLIN DD statement (z/OS) or the FILEDEF SYSLIN command (CMS). The generalized object format data set can only be processed by DFSMS/MVS 1.3 or later. This format requires the LIST=133 or LIST=MAX option.

NO the assembler does not produce a generalized object format data set.

Default
XOBJECT=NO

Notes:
1. This option has been superseded by the option GOFF.
2. See also “XOBJADATA (z/OS and CMS).”

XREF



SHORT
symbol cross-reference information for only those symbols that are referred to is generated as part of the assembler listing. Any symbols that are defined, but not referred to, are not included in the cross-reference listing.

FULL
symbol cross-reference information for all symbols used in the assembly is generated as part of the assembler listing. This includes symbols that are defined, but never referred to.

UNREFS
the assembler produces the *Unreferenced Symbols Defined in CSECTs* section of the assembler listing. The symbols are listed in symbol name order. UNREFS can be specified with the SHORT suboption to produce a cross-reference list of referenced symbols. The UNREFS suboption can not be specified with the FULL suboption.

NO symbol cross-reference information is not generated as part of the assembler listing.

Default
XREF=(SHORT,UNREFS)

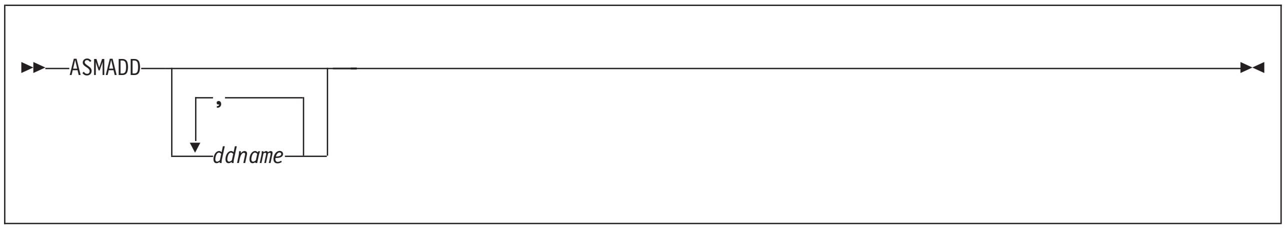
Appendix B. High Level Assembler DDNAMES (z/OS and CMS)

The z/OS and CMS installation macro, ASMADD, enables you to specify installation default DDnames for the assembler data sets, during installation of High Level Assembler.

Note: For further information about how to change the DDNames, using ASMADD, see "Step 3: Change default options and DDNAMES" on page 27 (z/OS), or "Changing option and DDNAMES defaults" on page 63 (CMS).

ASMADD Syntax

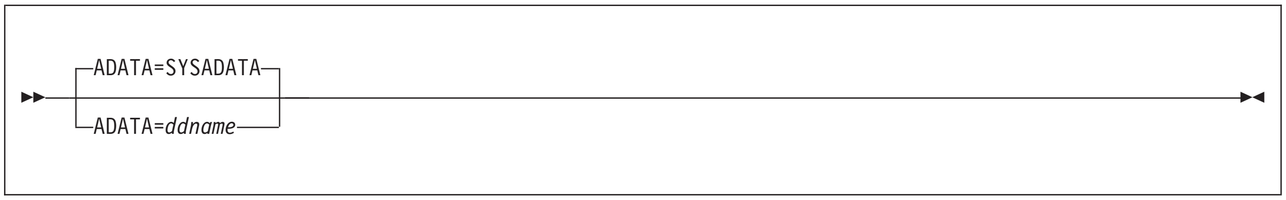
The syntax of the ASMADD installation macro is given below.



ASMADD

The operands of the ASMADD macro, and the values that can be specified for each operand, are described below. The IBM-supplied default values are shown above the main path.

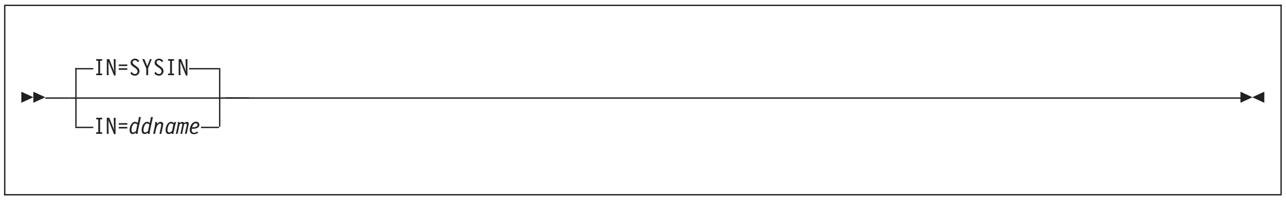
ADATA



ddname
: the *ddname* of the data set containing the assembler language program data that is generated by the assembler when the ADATA assembler option is specified.

Default
: ADATA=SYSADATA

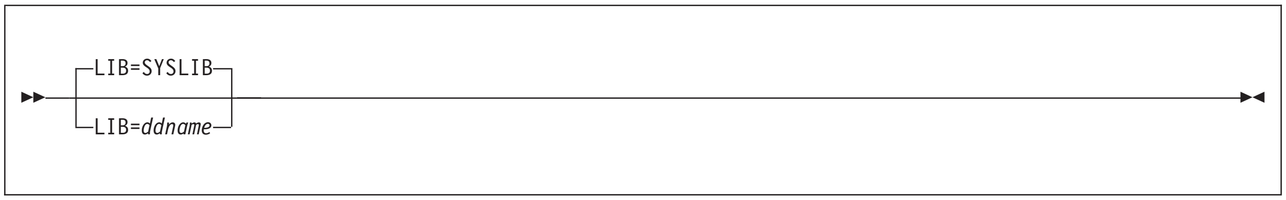
IN



ddname
: the *ddname* of the input data set containing source statements to be processed.

Default
: IN=SYSIN

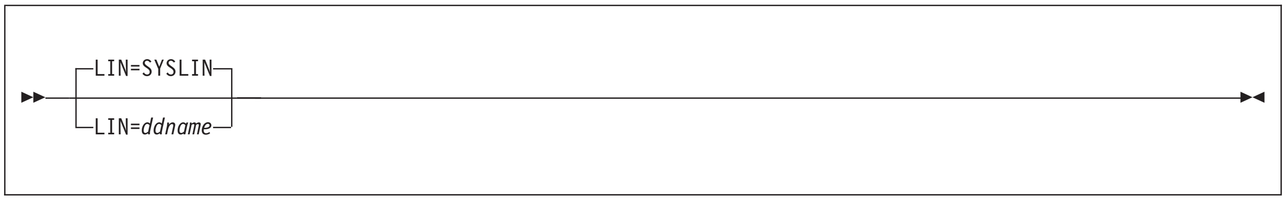
LIB



ddname
: the *ddname* of the data set containing system macros, installation-cataloged macros, and members to be included by COPY statements.

Default
: LIB=SYSLIB

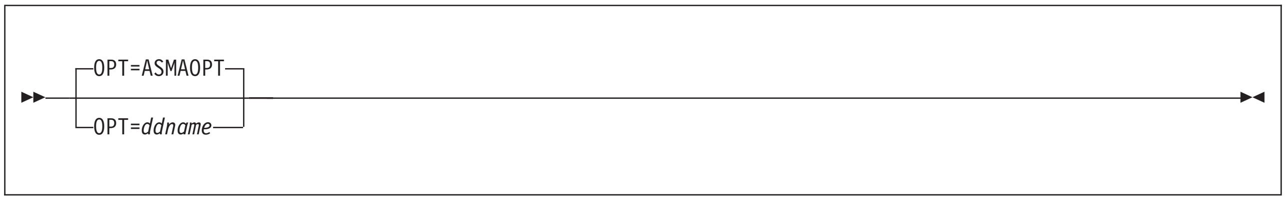
LIN



ddname
: the *ddname* of the data set containing the object module generated by the assembler when the OBJECT or XOBJECT assembler option is specified.

Default
: LIN=SYSLIN

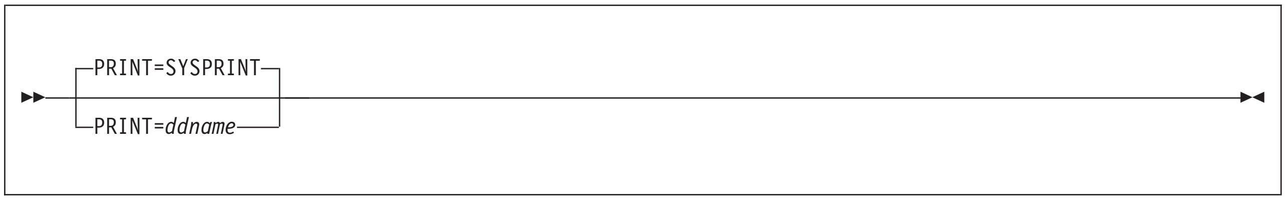
OPT



ddname
: the *ddname* of the input data set containing the assembler options.

Default
: OPT=ASMAOPT

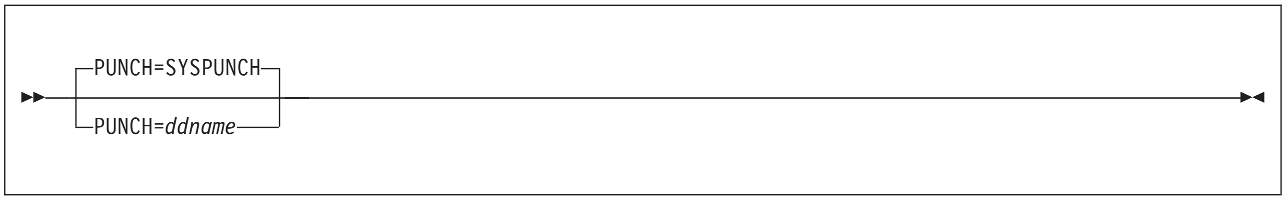
PRINT



ddname
: the *ddname* of the data set containing the assembler listing generated by the assembler when the LIST assembler option is specified.

Default
: PRINT=SYSPRINT

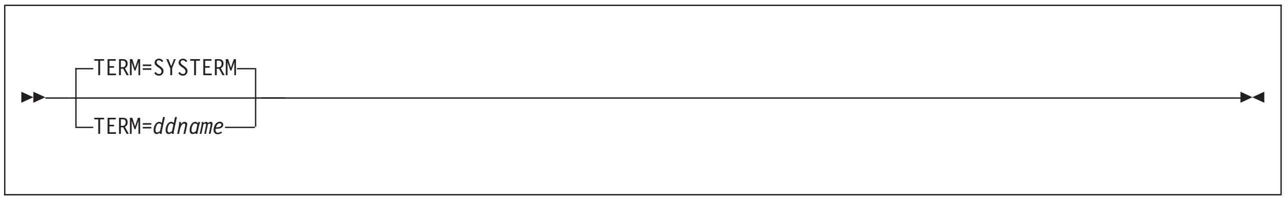
PUNCH



ddname
: the *ddname* of the data set containing the object module generated by the assembler when the DECK assembler option is specified.

Default
: PUNCH=SYSPUNCH

TERM



ddname
: the *ddname* of the data set containing error messages and assembler completion messages to be displayed at the terminal when the TERM assembler option is specified.

Default
: TERM=SYSTERM

UT1

```
      ┌─UT1=SYSUT1─┐
►►────┼────────────┼──────────────────────────────────────────────────────►◄
      └─UT1=ddname─┘
```

ddname
   the *ddname* of the data set used as the assembler work file.

Default
   UT1=SYSUT1

Appendix C. High Level Assembler Service

The following service for High Level Assembler Release 5 has been applied to Release 6, for all platforms including zLinux.

Table 51. Service history

APAR	z/OS PTF	VM PTF	z/VSE PTF
PQ88271	UQ88981	UQ88980	UQ88982
PQ88470	UQ94154	UQ94152	UQ94156
PQ89025	UQ89760	UQ89722	UQ89771
PQ89655	UQ89581	UQ89550	UQ89582
PQ90802	UQ90358	UQ90357	UQ90373
PQ91893	UK03034	UK03032	UK03036
PQ92291	UQ92091	UQ92084	UQ92093
PQ92371	UK02119	UK02118	UK02120
PQ92508	UK02957	UK02930	UK02977
PQ92579	UK00528	UK00527	UK00529
PQ93977	UQ93163	UQ93149	UQ93179
PQ95145	UQ96494	UQ96493	UQ96495
PQ96292	UK00998	UK00996	UK01000
PQ98607	UQ96787	UQ96786	UQ96788
PQ99158	UK06004	UK06002	UK06006
PQ99706	UK01050	UK01049	UK01051
PK00040	UK02619	UK02600	UK02621
PK01064	UK10289	UK10287	UK10291
PK02523	UK03372	UK03371	UK03373
PK02660	UK06525	UK06520	UK06527
PK05761	UK05928	UK05927	UK05929
PK06113	UK08101	UK08100	UK08102
PK06652	UK05446		
PK07828	UK27902	UK27901	UK27903
PK09700	UK13781	UK13780	UK13782
PK12545	UK10256	UK10254	UK10258
PK14299	UK09396	UK09392	UK09399
PK15306	UK16005	UK16004	UK16006
PK17439	UK10660	UK10656	UK10670
PK17447	UK10754	UK10752	UK10756
PK17728	UK13876	UK13874	UK13858
PK18170	UK14321	UK14319	UK14359
PK19083	UK19034	UK19033	UK19035
PK23005	UK15833	UK15830	UK15856

Table 51. Service history (continued)

APAR	z/OS PTF	VM PTF	z/VSE PTF
PK24143	UK15027	UK15028	UK15026
PK25298	UK14811	UK14809	UK14813
PK25410	UK15002	UK14991	UK15003
PK26756	UK17789	UK17788	UK17790
PK27282	UK17017	UK17016	UK17018
PK27577	UK16005	UK16004	UK16006
PK27657	UK16844	UK16843	UK16845
PK27979	UK17628	UK17627	UK17629
PK29624	UK18663	UK18661	UK18665
PK31383	UK18562	UK18561	UK18563
PK31465	UK18972	UK18970	UK18974
PK34746	UK28644	UK28643	UK28645
PK36579	UK21335	UK21334	UK21336
PK37014	UK23192	UK23190	UK23194
PK37093	UK21575	UK21571	UK21604
PK40237	UK24440	UK24439	UK24441
PK42535	UK27290	UK27288	UK27292
PK43179	UK24690	UK24689	UK24691
PK55677	UK32921	UK32920	UK32922
PK55678	UK31917	UK31916	UK31918
PK56245	UK34454	UK34453	UK34455
PK56672	UK33757	UK33750	UK33758
PK58463	UK33787	UK33764	UK33788

Appendix D. Create Product Parameter File (PPF) override (z/VM)

This section provides information to help you create a product parameter file (PPF) override. The example used in this section shows how to change the shared file system (SFS) file pool where High Level Assembler files reside.

Note: Do **not** modify the product supplied P696234J $PPF or P696234J PPF files to change the file pool name or any other installation parameters. If the P696234J $PPF file is serviced, the existing $PPF file will be replaced, and any changes to that file will be lost; by creating your own $PPF override, your updates will be preserved.

The following process describes changing the default file pool name, VMSYS, to MYPOOL1:

1. Create a new $PPF override file, or edit the override file created via the 'Make Override Panel' function.

Command	Explanation
xedit *overname* **$PPF** *fm***2**	*overname* is the PPF override file name (such as "myHLASM") that you want to use. *fm* is an appropriate file mode. If you create this file yourself, specify a file mode of A.

   If you modify an existing override file, specify a file mode of A or D, based on where the file currently resides (A being the file mode of a R/W 191 minidisk, or equivalent; D, that of the MAINT 51D minidisk).

2. Create (or modify as required) the Variable Declarations (:DCL.) section for the HLASMSFS override area, so that it resembles the :DCL. section shown below. This override is used for the installation of High Level Assembler.

```
:OVERLST. HLASMSFS
*
* ================================================================== *
* Override Section for Initial Installation (Using SFS Directories)   *
* ================================================================== *
:HLASMSFS. HLASMSFS P696234J
:DCL. REPLACE
&191     DIR MYPOOL1:P696234J.
&SAMPZ   DIR MYPOOL1:P696234J.HLASM.LOCAL
&DELTZ   DIR MYPOOL1:P696234J.HLASM.DELTA
&APPLX   DIR MYPOOL1:P696234J.HLASM.APPLYALT
&APPLZ   DIR MYPOOL1:P696234J.HLASM.APPLYPROD
&BLD0Z   DIR MYPOOL1:P696234J.HLASM.TBUILD
&BAS1Z   DIR MYPOOL1:P696234J.HLASM.OBJECT
&HLAID1  USER  P696234J
:EDCL.
:END.
*
```

   (This override replaces the :DCL. section of the HLASMSFS override area of the P696234J $PPF file.)

3. If your $PPF override file was created on the A disk, copy it to file mode D—the Software Inventory minidisk (MAINT 51D). Then erase it from file mode A.

Command	Explanation
copyfile *overname* **$PPF** *fm* **= = d (olddate**	

Command	Explanation
	Move PPF file to software inventory disk.

erase *overname* **$PPF** *fm*	Erase redundant file on *fm* disk.

Options:

OLDDATE
Use the date and time on the input file as the date and time of last update of the output file.

4. Compile your changes to create the usable *overname* PPF file.

Command	Explanation
vmfppf *overname* **HLASMSFS**	*overname* is the file name of your $PPF override file.

Now that the *overname* PPF file has been created, you should specify *overname* instead of P696234J as the PPF name for those VMSES/E commands that require a PPF name.

Appendix E. Local modification procedures (z/VM)

If a local modification exists for ASMADOPT or one is planned then the following should be followed.

Full part replacement local modification

To be done at initial install or first time a local modification is made to the ASMADOPT ASSEMBLE file.

1. Establish HLASM minidisk order

   VMFSETUP P696234J {HLASM | HLASMSFS}
2. Build a new copy of the MACLIB (This is needed so you get the current copy of any macros that the assemble file needs.)

   VMFBLD PPF P696234J {HLASM | HLASMSFS} ASMBLMAC (SERVICED
3. Make a copy of the assemble file on the 2C2 (E-disk) using the local modid as part of the file type.

   COPYFILE ASMADOPT ASSEMBLE *fm-2c2* **= ASML***nnnn fm-2c2*

   Note: ASML is a required file type for local modifications. The nnnn is a user-defined number assigned to this fix.
4. Incorporate your local modification into the copy of the assemble (ASML*nnnn*) file on your LOCALSAM 2C2 disk.
5. Update the VVT tables for both the TXT and ASM files by issuing the following VMFSIM commands.

   VMFSIM LOGMOD P696234J VVTLCL E TDATA :MOD LCL*nnnn* **:PART ASMADOPT TXT**

   VMFSIM LOGMOD P696234J VVTLCL E TDATA :MOD LCL*nnnn* **:PART ASMADOPT ASM**
6. Build the new assemble file to be used in next step

   VMFBLD PPF P696234J {HLASM | HLASMSFS} ASMBLSAM ASMADOPT (ALL
7. Issue the assemble command for the file.

   VMFHLASM ASMADOPT P696234J {HLASM | HLASMSFS} ($SELECT OUTMODE E NOCKGEN HLASM NOSEG EHLASM

   Note: Consult VMSES/E Introduction and Reference for additional information, for extra ASSEMBLE options.

   If the assemble function is successful, then file ASMADOPT TXTL*nnnn* will be placed on the LOCALSAM 2C2 (E) disk
8. Build your local modification on the test build disk by issuing the following command.

   VMFBLD PPF P696234J {HLASM | HLASMSFS} (SERVICED
9. Place the local modification into production

   Typically this is placing the ASMADOPT module on MAINT's 19E disk. Use VMFCOPY from within MAINT's userid.

Reworking local modification

To rework a full part replacement local modification to the ASMADOPT ASSEMBLE file do the following.

1. Establish HLASM minidisk order

   VMFSETUP P696234J {HLASM | HLASMSFS}

2. If you are re-working your local modification to an assemble file because the assemble file was just serviced then you need to find the highest level of IBM service to the part because you will need to compare the IBM serviced level to your local mod'ed level. To do this, you need to issue a VMFSIM GETLVL against just that assemble file.

   VMFSIM GETLVL P696234J {HLASM | HLASMSFS} TDATA :PART ASMADOPT ASM (HISTORY

   The command returns information to determine the highest level of this part as shipped by IBM. If the response contains the :PTF tag, then there is IBM service to the part. The highest level of IBM service to the part is fn ftabbrev-ptfnumber. Use these names for the file name and file type of the highest level of the part. If the response does not contain the :PTF tag, then there is no IBM service to the part.
3. Compare your local modification on the LOCALSAM 2C2 disk with the highest level of IBM service. Make any changes necessary.
4. If there were any changes made to your local mod then build a new copy of the ASSEMBLE file.

   VMFBLD PPF P696234J {HLASM | HLASMSFS} ASMBLSAM ASMADOPT (ALL
5. If you are re-working your local modification to an assemble file because the macro or text file was serviced that this assemble file uses then you need to rebuild the new macro or MACLIB, if it is in one, before during the assembly. To do this you need to issue a VMFBLD command.

   VMFBLD PPF P696234J {HLASM | HLASMSFS} ASMBLMAC (SERVICED
6. Issue the assemble command for the file.

   VMFHLASM ASMADOPT P696234J {HLASM | HLASMSFS} ($SELECT OUTMODE E HLASM NOSEG EHLASM

   Note: Consult *z/VM: VMSES/E Introduction and Reference* for additional information, for extra ASSEMBLE options.

   If the assemble function is successful, then file ASMADOPT TXTL*nnnn* will be placed on the LOCALSAM 2C2 (E) disk

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie New York 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

Trademarks

Bibliography

High Level Assembler Documents

HLASM General Information, GC26-4943

HLASM Installation and Customization Guide, SC26-3494

HLASM Language Reference, SC26-4940

HLASM Programmer's Guide, SC26-4941

Toolkit Feature document

HLASM Toolkit Feature User's Guide, GC26-8710

HLASM Toolkit Feature Debug Reference Summary, GC26-8712

HLASM Toolkit Feature Interactive Debug Facility User's Guide, GC26-8709

HLASM Toolkit Feature Installation and Customization Guide, GC26-8711

Related documents (Architecture)

z/Architecture Principles of Operation, SA22-7832

Related documents for z/OS

z/OS:

z/OS MVS JCL Reference, SA23-1385

z/OS MVS JCL User's Guide, SA23-1386

z/OS MVS Programming: Assembler Services Guide, SA23-1368

z/OS MVS Programming: Assembler Services Reference, Volume 1 (ABE-HSP), SA23-1369

z/OS MVS Programming: Assembler Services Reference, Volume 2 (IAR-XCT), SA23-1370

z/OS MVS Programming: Authorized Assembler Services Guide, SA23-1371

z/OS MVS Programming: Authorized Assembler Services Reference, Volumes 1 - 4, SA23-1372 - SA23-1375

z/OS MVS Program Management: User's Guide and Reference, SA23-1393

z/OS MVS System Codes, SA38-0665

z/OS MVS System Commands, SA38-0666

z/OS MVS System Messages, Volumes 1 - 10, SA38-0668 - SA38-0677

z/OS Communications Server: SNA Programming, SC27-3674

UNIX System Services:

z/OS UNIX System Services User's Guide, SA23-2279

DFSMS/MVS:

z/OS DFSMS Program Management, SC27-1130

z/OS DFSMSdfp Utilities, SC23-6864

TSO/E (z/OS):

z/OS TSO/E Command Reference, SA32-0975

SMP/E (z/OS):

SMP/E for z/OS Messages, Codes, and Diagnosis, GA32-0883

SMP/E for z/OS Reference, SA23-2276

SMP/E for z/OS User's Guide, SA23-2277

Related documents for z/VM

z/VM: VMSES/E Introduction and Reference, GC24-6243

z/VM: Service Guide, GC24-6247

z/VM: CMS Commands and Utilities Reference, SC24-6166

z/VM: CMS File Pool Planning, Administration, and Operation, SC24-6167

z/VM: CP Planning and Administration, SC24-6178

z/VM: Saved Segments Planning and Administration, SC24-6229

z/VM: Other Components Messages and Codes, GC24-6207

z/VM: CMS and REXX/VM Messages and Codes, GC24-6161

z/VM: CP System Messages and Codes, GC24-6177

z/VM: CMS Application Development Guide, SC24-6162

z/VM: CMS Application Development Guide for Assembler, SC24-6163

z/VM: CMS User's Guide, SC24-6173

z/VM: XEDIT User's Guide, SC24-6245

z/VM: XEDIT Commands and Macros Reference, SC24-6244

z/VM: CP Commands and Utilities Reference, SC24-6175

Related documents for z/VSE

z/VSE: Guide to System Functions, SC33-8312

z/VSE: Administration, SC34-2627

z/VSE: Installation, SC34-2631

z/VSE: Planning, SC34-2635

z/VSE: System Control Statements, SC34-2637

z/VSE: Messages and Codes, Vol.1 , SC34-2632

z/VSE: Messages and Codes, Vol.2, SC34-2633

z/VSE: Messages and Codes, Vol.3, SC34-2634

REXX/VSE Reference, SC33-6642

REXX/VSE User's Guide, SC33-6641

Glossary

This glossary defines terms and abbreviations that are used in this book. If you do not find the term you are looking for refer to the index, to the glossary of the appropriate high-level language (HLL) manual, or to the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

A

abend Abnormal end of application.

accept An SMP/E process that moves distributed code and programs to the distribution libraries.

activate
To make a program available for use.

addressing mode (AMODE)
An attribute that refers to the address length that a routine is prepared to handle upon entry. Addresses may be 24 or 31 bits long.

address space
Domain of addresses that are accessible by an application.

AMODE
Addressing mode.

APAR Authorized program analysis report.

authorized program analysis report (APAR)
A request for correction of a problem caused by a defect in a current release of a program.

authorized program facility (APF)
The authorized program facility (APF) is a facility that an installation manager uses to protect the system. In MVS, certain system functions, such as all or part of some SVCs, are sensitive; their use must be restricted to users who are authorized. An authorized program is one that executes in supervisor state, or with APF authorization.

auxiliary file
In CMS, a file that contains a list of file types of update files to be applied to a particular source file.

B

base The core product, upon which features may be separately ordered and installed.

batch Pertaining to activity involving little or no user action. Contrast with *interactive*.

byte The basic unit of storage addressability, normally with a length of 8 bits.

C

CBIPO
Custom-Built Installation Process Offering.

CBPDO
Custom-Built Product Delivery Offering.

CE IBM customer engineer.

CLIST TSO command list.

CMS Conversational monitor system.

compiler options
Keywords that can be specified to control certain aspects of compilation. Compiler options can control the nature of the load module generated by the compiler, the types of printed output to be produced, the efficient use of the compiler, and the destination of error messages.

component
Software that is part of a functional unit.

A set of modules that performs a major function within a system.

condition code
A code that reflects the result of a previous input/output, arithmetic, or logical operation.

control block
A storage area used by a computer program to hold control information.

control file
In CMS, a file that contains records that identify the updates to be applied and the macrolibraries, if any, needed to assemble a particular source program.

control program (CP)
A computer program designed to schedule and to supervise the execution of programs of a computer system.

control section (CSECT)
The part of a program specified by the programmer to be a relocatable unit, all

elements of which are to be loaded into adjoining main storage locations.

control statement
In programming languages, a statement that is used to alter the continuous sequential execution of statements; a control statement can be a conditional statement, such as IF, or an imperative statement, such as STOP.

In JCL, a statement in a job that is used in identifying the job or describing its requirements to the operating system.

conversational monitor system (CMS)
A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities, and operates only under the control of the VM/370 control program.

corrective maintenance
Maintenance performed specifically to overcome existing problems.

CP command
In VM, a command by which a terminal user controls his or her virtual machine. The VM/370 control program commands are called CP commands.

CPPL Command processor parameter list.

CP privilege class
In VM, one or more classes assigned to a virtual machine user in the user's VM directory entry; each privilege class allows access to a logical subset of the CP commands.

CSI Consolidated software inventory data set. See *SMPCSI*.

CSECT
Control section.

cumulative service tape
A tape sent with a new function order, containing all current PTFs for that function.

Custom-Built Installation Process Offering (CBIPO)
A CBIPO is a tape that has been specially prepared with the products (at the appropriate release levels) requested by the customer. A CBIPO simplifies installing various products together.

Custom-Built Product Delivery Offering (CBPDO)
A CBPDO is a tape that has been specially prepared for installing a particular product and the related service requested by the customer. A CBPDO simplifies installing a product and the service for it.

D

data definition name (DDNAME)
The logical name of a file within an application. The DDNAME provides the means for the logical file to be connected to the physical file.

data set
On MVS, a named collection of related data records that is stored and retrieved by an assigned name. Equivalent to a CMS *file*.

data set name (dsname)
The data set name on the DD statement in the JCL or the dsname operand of the TSO ALLOC command.

DBCS Double-byte character set.

DDDEF
Dynamic data definition.

DDNAME
Data definition name.

default
A value that is used when no alternative is specified.

DD statement
In MVS, connects the logical name of a file and the physical name of the file.

DELTA disk
In VM, the virtual disk that contains program temporary fixes (PTFs) that have been installed but not merged.

distribution libraries
IBM-supplied partitioned data sets on tape containing one or more components that the user restores to disk for subsequent inclusion in a new system.

distribution medium
The medium on which software is distributed to the user; for example, 9-track magnetic tape, tape cartridge.

distribution zone
In SMP/E, a group of VSAM records that

describe the SYSMODs and elements in the distribution libraries.

DITTO utility
Data Interfile Transfer, Testing, and Operations utility.

double-byte character set (DBCS)
A collection of characters represented by a 2-byte code.

driving system
The system used to install the program. Contrast with target system.

dsname
Data set name.

dynamic data definition (DDDEF)
The process of defining a data set and allocating auxiliary storage space for it while, rather than before, a job step executes.

dynamic storage
Storage acquired as needed at run time. Contrast with *static storage*.

E

ECMODE
Extended control mode.

executable program
A program that has been link-edited and therefore can run in a processor.

The set of machine language instructions that constitute the output of the compilation of a source program.

Extended control mode (ECMODE)
A mode in which all features of a System/370 computing system, including dynamic address translation, are operational.

Extended Service Option (ESO)
A service option that gives a customer all the new fixes for problems in IBM licensed programs that operate under that customer's operating system.

F

feature
A part of an IBM product that may be ordered separately by a customer.

feature number
A four-digit code used by IBM to process hardware and software orders.

file A named collection of related data records that is stored and retrieved by an assigned name. Equivalent to an MVS *data set*.

FILEDEF
File definition statement.

file definition statement (FILEDEF)
In CMS, connects the logical name of a file and the physical name of a file.

fix A correction of an error in a program, normally a temporary correction or bypass of defective code.

FMID Function modification identifier.

Fortran
A high-level language primarily designed for applications involving numeric computations.

function
A routine that is invoked by coding its name in an expression. The routine passes a result back to the invoker through the routine name.

function modification identifier (FMID)
The value used to distinguish separate parts of a product. A product tape or cartridge has at least one FMID.

I

IBM customer engineer (CE)
An IBM service representative who performs maintenance services for IBM hardware.

IBM program support representative (PSR)
An IBM service representative who performs maintenance services for IBM software at a centralized IBM location.

IBM service representative
An individual in IBM who performs maintenance services for IBM products or systems.

IBM Software Distribution (ISD)
The IBM department responsible for software distribution.

IBM Support Center
The IBM department responsible for software service.

IBM systems engineer (SE)
An IBM service representative who performs maintenance services for IBM software in the field.

initial program load (IPL)
The initialization procedure that causes an operating system to commence operation.

The process by which a configuration image is loaded into storage, as at the beginning of a work day or after a system malfunction or as a means to access updated parts of the system.

The process of loading system programs and preparing a system to run jobs.

inline Sequential execution of instructions, without branching to routines, subroutines, or other programs.

IPL Initial program load.

interactive
Pertaining to a program or system that alternately accepts input and responds. In an interactive system, a constant dialog exists between user and system. Contrast with *batch*.

Interactive Interface
A series of panels, allowing the user to use the facilities of the VSE/ESA operating. This interface runs within CICS®/VSE.

Interactive System Productivity Facility (ISPF)
ISPF is a dialog manager for interactive applications. It provides control and services to permit execution of dialogs.

ISD IBM Software Distribution.

J

JCL Job control language.

JCLIN data
The JCL statements associated with the ++JCLIN statement or saved in the SMPJCLIN data set. They are used by SMP/E to update the target zone when the SYSMOD is applied. Optionally, SMP/E can use the JCLIN data to update the distribution zone when the SYSMOD is accepted.

JES Job Entry Subsystem

Job Entry Subsystem
A system facility for spooling, job queueing, and managing the scheduler work area.

job control language (JCL)
A sequence of commands used to identify a job to an operating system and to describe a job's requirements.

job step
You enter a program into the operating system as a job step. A job step consists of the job control statements that request and control execution of a program and request the resources needed to run the program. A job step is identified by an EXEC statement. The job step can also contain data needed by the program. The operating system distinguishes job control statements from data by the contents of the record.

L

librarian
In VSE, the set of programs that maintains, services, and organizes the system and private libraries.

library
A collection of functions, subroutines, or other data.

link pack area (LPA)
In MVS, an area of main storage containing reenterable routines from system libraries. Their presence in main storage saves loading time when a reenterable routine is needed.

linkage editor
A program that resolves cross-references between separately assembled object modules and then assigns final addresses to create a single relocatable load module. The linkage editor then stores the load module in a program library in main storage.

link-edit
To create a loadable computer program by means of a linkage editor.

load module
An application or routine in a form suitable for execution. The application or routine has been compiled and link-edited; that is, address constants have been resolved.

logical saved segment
A portion of a physical saved segment that CMS can manipulate. Each logical saved segment can contain different types of program objects, such as modules, text files, execs, callable services libraries,

language repositories, user-defined objects, or a single minidisk directory. A system segment identification file (SYSTEM SEGID) associates a logical saved segment to the physical saved segment in which it resides. See *physical saved segment* and *saved segment*.

LPA Link pack area.

M

maintain system history program (MSHP)
In VSE, a program used for automating and controlling various installation, tailoring, and service activities for a VSE system.

MCS Modification control statement

minidisk
In VM, all, or a logical subdivision of, a physical disk storage device that has its own address, consecutive storage space for data, and an index or description of stored data so that the data can be accessed. Synonymous with virtual disk.

module
A language construct that consists of procedures or data declarations and can interact with other such constructs.

MSHP
Maintain system history program.

MVS Multiple Virtual Storage operating system.

multicultural support
Translation requirements affecting parts of licensed programs; for example, translation of message text and conversion of symbols specific to countries.

N

Named Saved System
A copy of an operating system that a user has named and saved in a file. The user can load the operating system by its name, which is more efficient than loading it by device number.

nonexecutable components
Components of a product that cannot be run.

non reentrant
A program that cannot be shared by multiple users.

nonreenterable
See *non reentrant*.

NSS named saved system

O

object code
Output from a compiler or assembler which is itself executable machine code or is suitable for processing to produce executable machine code.

object deck
Synonymous with *object module, text deck*.

object module
A portion of an object program suitable as input to a linkage editor. Synonymous with *text deck, object deck*.

online Pertaining to a user's ability to interact with a computer.

Pertaining to a user's access to a computer via a terminal.

operating system
Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

P

parameter
Data items that are received by a routine.

partition
A fixed-size division of storage.

phase In VSE, the smallest complete unit of executable code that can be loaded into virtual storage.

physical saved segment
One or more pages of storage that have been named and retained on a CP-owned volume (DASD). When created, it can be loaded within a virtual machine's address space or outside a virtual machine's address space. Multiple users can load the same copy. A physical saved segment can contain one or more logical saved segments. A system segment identification file (SYSTEM SEGID) associates a physical saved segment to its logical saved segments. See *logical saved segment* and *saved segment*.

preventive maintenance
Maintenance performed specifically to prevent problems from occurring.

preventive service planning (PSP)
The online repository of program temporary fixes (PTFs) and other service information. This information could affect installation.

procedure
A named block of code that can be invoked, normally with a call.

procedure library (PROCLIB)
A program library in direct-access storage with job definitions. The reader/interpreter can be directed to read and interpret a particular job definition by an execute statement in the input stream.

PROCLIB
Procedure library.

program level
The modification, release, version, and fix level of a product.

program number
The seven-digit code (in the format *xxxx-xxx*) used by IBM to identify each program product.

program temporary fix (PTF)
A temporary solution or bypass of a problem diagnosed by IBM as resulting from a defect in a current unaltered release of the program.

PSP Preventive service planning.

PSR IBM program support representative.

PTF Program temporary fix.

PWS Programmable workstation.

Q

qualifier
A modifier that makes a name unique.

R

reentrant
The attribute of a routine or application that allows more than one user to share a single copy of a load module.

reenterable
See *reentrant*

relative file tape (RELFILE tape)
A standard label tape made up of two or more files. It contains a file of the MCSs for one or more function SYSMODs and one or more relative files containing unloaded source data sets and unloaded, link-edited object data sets at the distribution library level. A relative file tape is one way of packaging SYSMODs, and is typically used for function SYSMODs.

relative files (RELFILEs)
Files containing modification text and JCL input data associated with a SYSMOD.

RELFILEs
Relative files

RELFILE tape
Relative file tape

relocatable load module
On CMS, a combination of object modules having cross references resolved and prepared for loading into storage for execution.

residence mode (RMODE)
The attribute of a load module that specifies whether the module, when loaded, must reside below the 16MB virtual storage line or may reside anywhere in virtual storage.

resident modules
A module that remains in a particular area of storage.

return code
A code produced by a routine to indicate its success. It can be used to influence the execution of succeeding instructions.

RIM Related installation materials

RMODE
Residence mode.

run To cause a program, utility, or other machine function to be performed.

S

save area
Area of main storage in which contents of registers are saved.

SBCS Single-byte character set.

SE IBM systems engineer.

service level
The modification level, release, version,

and fix level of a program. The service level incorporates PTFs if there are any.

saved segment
A segment of storage that has been saved and assigned a name. Saved segments can be physical saved segments that CP recognizes or logical saved segments that CMS recognizes. The segments can be loaded and shared among virtual machines, which helps use real storage more efficiently, or a private, nonshared copy can be loaded into a virtual machine. See *logical saved segment* and *physical saved segment*.

shared segment
In VM, a feature of a saved system that allows one or more segments of reenterable code in real storage to be shared among many virtual machines.

shared storage
An area of storage that is the same for each virtual address space. Because it is the same space for all users, information stored there can be shared and does not have to be loaded in the user region.

shared virtual area (SVA)
In VSE, a high address area of virtual storage that contains a system directory list (SDL) of frequently used phases, resident programs that can be shared between partitions, and an area for system support.

severity code
A part of run-time messages that indicates the severity of the error condition (1, 2, 3, or 4).

single-byte character set (SBCS)
A collection of characters represented by a 1-byte code.

SMPCSI
The SMP/E data set that contains information about the structure of a user's system as well as information needed to install the operating system on a user's system. The SMPCSI DD statement refers specifically to the CSI that contains the global zone. This is also called the master CSI.

softcopy
One or more files that can be electronically distributed, manipulated, and printed by a user.

software inventory disk
In VM, the disk where the system level inventory files reside.

source code
The input to a compiler or assembler, written in a source language.

source program
A set of instructions written in a programming language that must be translated to machine language before the program can be run.

SREL System release identifier

statement
In programming languages, a language construct that represents a step in a sequence of actions or a set of declarations.

sublibrary
In VSE, a subdivision of a library.

SUBSET
The value that specifies the function modifier (FMID) for a product level. It further specifies an entry in RETAIN for a product level.

subsystem
A secondary or subordinate system, or programming support, normally capable of operating independently of or asynchronously with a controlling system. Examples are CICS and IMS.

SVA Shared virtual area.

syntax The rules governing the structure of a programming language and the construction of a statement in a programming language.

SYSMOD
system modification.

SYSMOD ID
system modification identifier.

system abend
An abend caused by the operating system’s inability to process a routine; can be caused by errors in the logic of the source routine.

T

target disk
In VM, the disk to which a program is installed.

target libraries
In SMP/E, a collection of data sets in which the various parts of an operating system are stored. These data sets are sometimes called system libraries.

target zone
In SMP/E, a collection of VSAM records describing the target system macros, modules, assemblies, load modules, source modules, and libraries copied from DLIBs during system generation, and the system modifications (SYSMODs) applied to the target system.

text deck
Synonym for *object module, object deck*.

time sharing option/extended (TSO/E)
An option on the operating system; for System/370, the option provides interactive time sharing from remote terminals.

TSO/E Time sharing option/extended.

U

UCLIN
In SMP/E, the command used to initiate changes to SMP/E data sets. Actual changes are made by subsequent UCL statements.

UPGRADE
An alphanumeric identifier that specifies a product level.

user exit
A routine that takes control at a specific point in an application.

USERMOD
User modification.

user modification (USERMOD)
A change to product code that the customer initiates.

V

virtual machine (VM)
A functional simulation of a computer and its associated devices. Each virtual machine is controlled by a suitable operating system.

In VM, a functional equivalent of either a System/370 computing system or a System/370-Extended Architecture computing system.

VMFINS
An installation aid supplied as part of VMSES/E to make installation on VM consistent.

VM Serviceability Enhancements Staged/Extended (VMSES/E)
A program product for installing and maintaining products on VM.

VMSES/E
VM Serviceability Enhancements Staged/Extended.

VOLSER
Volume serial number.

volume
A certain portion of data, together with its data carrier, that can be handled conveniently as a unit.

A data carrier mounted and demounted as a unit; for example, a reel of magnetic tape, a disk pack.

volume label
An area on a standard label tape used to identify the tape volume and its owner. This area is the first 80 bytes and contains VOL 1 in the first four positions.

volume serial number (VOLSER)
A number in a volume label assigned when a volume is prepared for use in a system.

VSAM
Virtual storage access method. A high-performance mass storage access method. Three types of data organization are available: entry sequenced data sets (ESDS), key sequenced data sets (KSDS), and relative record data sets (RRDS).

Index

SC26-3494-05