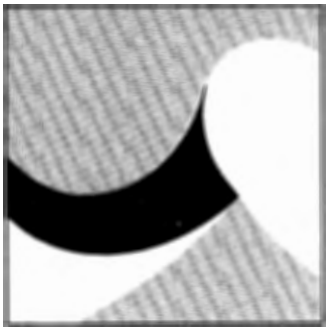
IBM

Network Control Program and System Support Programs Generation and Loading Guide

SC30-3348-0

Network Control Program and System Support Programs Generation and Loading Guide



Advanced Communications Function for Network Control Program
Versions 3 and 4 Program Numbers: 5667-124, 5668-854

Advanced Communications Function for System Support Programs
Version 3 Program Numbers: 5665-338 (MVS), 5666-322 (VSE), 5664-289 (VM)

Advanced Communications Function for Network Control Program
V4 Subset Program Number: 5668-754

SC30-3348-0

File No. S370/4300/30XX-50

First Edition (May 1986)

This edition applies to:

Advanced Communications Function for Network Control Program Version 3, Program Product 5667-124

Advanced Communications Function for Network Control Program Version 4, Program Product 5668-854

Advanced Communications Function for Network Control Program V4 Subset, Program Product 5668-754

Advanced Communications Function for System Support Programs Version 3, Program Products:

5665-338 (for MVS)
5666-322 (for VSE)
5664-289 (for VM/SP).

This edition also applies to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30XX and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department E03, P.O. Box 12195, Research Triangle Park, North Carolina, U.S.A. 27709. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

NCP, SSP, and EP Library Overview

The diagrams on the following pages will help you find the NCP, SSP, or EP books that contain the information you need.

Migration and Generation

For Information About:	*Look In:*	*Order Number:*
How to generate an NCP/PEP load module How to load an NCP/PEP load module into your communication controller	NCP and SSP Generation and Loading Guide	SC30-3348
How to change, delete, or add definition statements and operands in your NCP generation definition if you are migrating from a previous release of NCP to a new release of NCP	NCP and SSP Migration	SC30-3252
Which NCP definition statements and operands to code for your network	NCP and SSP Resource Definition Guide	SC30-3349
How to code NCP definition statements and operands	NCP and SSP Resource Definition Reference	SC30-3254
Which EP definition statements and operands to code for your network How to code EP definition statements and operands How to generate an EP load module How to load an EP load module into your controller	EP Installation, Resource Definition, and Diagnosis	SC30-3338

Diagnosis and Customization

For Information About:	*Look In:*	*Order Number:*
User-written code NCP internal macros Structured coding, timer routines, user exits, block handling routines, machine language coding tips Customizing your NCP for line control Standard attachment facility	NCP Customization	LY30-5571
NCP diagnostic procedures Determining if you have an NCP problem Determining what type of NCP problem you are having Which traces and dumps to gather for an error How to run the SSP utilities or the Configuration Report Program for NCP How to prepare an APAR Determining the correct level of IBM support	NCP and SSP Diagnosis Guide	LY30-5591
The normal functioning and structure of the SSP program and utilities	SSP Diagnosis Reference	LY30-5564
Determining if you have an EP problem Determining what type of EP problem you have	EP Installation, Resource Definition, and Diagnosis	SC30-3338
EP error responses and ABEND codes The normal functioning and structure of the EP program EP, BSC and start stop commands, line control, save areas, line testing, channel responses for EP EP modules: function and entry points	EP Logic	LY30-3195
SSP messages or NCP ABEND codes	NCP and SSP Messages and Codes	SC30-3169
NCP error responses The normal functioning and structure of the NCP program NCP and start-stop commands, line control, save areas, line testing, channel responses	NCP Reference	LY30-5569
Data area relationships NCP control blocks BTU commands and modifiers, NCP channel commands, NCP network commands EP command codes Controller instruction set RECMS, RECFMS, NMVT, and NPM formats SVC codes, line character codes Character mode PCF diagrams NCP exception responses	NCP and EP Reference Summary and Data Areas	LY30-5570

About This Manual

The following contains information you should know before using this manual.

What This Manual is For

This manual will help you generate and load the following versions of NCP:

- Advanced Communications Function for Network Control Program (NCP) Version 3 for the IBM 3705 Communications Controller
- Advanced Communications Function for Network Control Program (NCP) Version 3 for the IBM 3725 Communication Controller
- Advanced Communication Function for Network Control Program (NCP) Version 4
- Advanced Communication Function for Network Control Program (NCP) V4 Subset

Note: Throughout this manual, the term "macro" (used in previous releases of the NCP and SSP manuals) has been replaced by the term "definition statement" and the term "stage" has been replaced by the term "phase."

This manual covers two of the basic steps for producing an operating NCP for your user-application network: generating and loading.

Note: The term ***network*** has at least two meanings. A ***public network*** is a network established and operated by communication common carriers or telecommunication administrations for the specific purpose of providing circuit-switched, packet-switched, and leased-circuit services to the public. A ***user-application network*** is a configuration of data processing products, such as processors, controllers, and terminals, established and operated by users for data processing or information exchange, which may use services offered by communication common carriers or telecommunication Administrations. ***Network***, as used in this manual, refers to a user-application network.

How to Use This Manual

This manual is designed to help you understand the generation and loading procedures and to help you determine the control statements you need to supply to generate and load your NCP.

Note: Throughout this manual, the term "your NCP" refers to the particular NCP you are generating and loading.

Before you generate and load your NCP, you must define to your NCP the resources in your network. For more information about defining your NCP, see the *ACF/NCP-SSP Resource Definition Guide* and the *ACF/NCP-SSP Resource Definition Reference* for descriptions of how to code the definition statements and operands. (The full title and order numbers of these manuals are given later under "Corequisite Manuals.")

Also before using this manual, you need to know the possible hardware and software combinations for SSP Version 3. The following table explains these:

NCP	Runs in This Controller	With This EP for PEP	Generated by These Operating Systems	Using This Version and Release of SSP
V3 for 3705	3705	EP/3705	MVS, VSE, VM/SP	SSP V3R1 or V3R2
V3 for 3725	3725	EP/3725 R2	MVS, VM/SP	SSP V3R1 or V3R2
V4R1	3725	EP/3725 R3	MVS, VSE	SSP V3R1 or V3R2
V4R1	3720	Release 3	MVS, VSE	SSP V3R2
V4R2	3725 or 3720	Release 4	MVS, VM/SP	SSP V3R2
V4 Subset	3720	Release 3	VSE	SSP V3R2
V4 Subset	3720	Release 4	VM/SP, MVS	SSP V3R2

Note: Throughout this manual, the use of the term "VM/SP" assumes that you are running under the VM/SP system in the CMS environment.

When you are ready to generate and load your NCP, locate the part of this book that covers the operating system under which you are generating and loading. The part of the book covering your operating system (1) gives you information on the generation and loading procedures, (2) tells you which control statements you need to supply, and (3) provides examples of control statements needed to generate and load your NCP.

How This Manual is Organized

This manual has four parts. These four parts include nine chapters, a glossary, and an index.

In this manual, any differences between versions and releases of NCP or SSP are noted. Otherwise, assume that the information applies to all versions and releases of NCP and SSP.

Parts, chapters, and appendixes in this manual are titled and organized as follows:

"Part One: Generating and Loading Under MVS": This part of the manual covers generating and loading under the MVS operating system. It contains Chapter 1, Chapter 2, and Chapter 3:

- **Chapter 1, "Generating the Program Under MVS":** This chapter explains what you need to know to generate your NCP under the MVS operating system. It describes the generation procedure under the MVS operating system, tells you what you need to code in the job control language to generate your NCP, and describes the listings you receive from the generation procedure.

- **Chapter 2, "Examples of Job Control Language for Generation under MVS":** This chapter contains examples of job control language for generating your NCP under the MVS operating system. It contains examples for the following types of generations:

  - A FASTRUN generation
  - An NCP/PEP generation with output written to tape
  - An NCP/PEP generation with output written to disk
  - An NCP/PEP generation with user-written code using the NDF standard attachment facility
  - An NCP/PEP generation with user-written code without using the NDF standard attachment facility
  - A dynamic reconfiguration generation.

- **Chapter 3, "Loading the Program Under MVS":** This chapter explains how to load your NCP into your communication controller under the MVS operating system. It describes the loader utility and explains the input you must provide to and the output you receive from the loader utility.

"Part Two: Generating and Loading Under VM/SP": This part of the manual covers generating and loading under the VM/SP operating system. It contains Chapter 4, Chapter 5, and Chapter 6:

- **Chapter 4, "Generating the Program Under VM/SP":** This chapter explains what you need to know to generate your NCP under the VM/SP operating system. It describes the generation procedure under the VM/SP operating system, tells you what you need to code in the EXECs to generate your NCP, and describes the listings you receive from the generation procedure.

- **Chapter 5, "Examples of EXECs for Generation under VM/SP":** This chapter contains examples of EXECs written in the REXX language for generating your NCP under the VM/SP operating system. It contains examples for the following types of generations:

  - A FASTRUN generation
  - An NCP/PEP generation with output written to tape
  - An NCP/PEP generation with output written to disk
  - An NCP/PEP generation with user-written code using the NDF standard attachment facility
  - An NCP/PEP generation with user-written code without using the NDF standard attachment facility
  - A dynamic reconfiguration generation.

- **Chapter 6, "Loading the Program Under VM/SP":** This chapter explains how to load your NCP into your communication controller under the VM/SP operating system. It describes the loader utility and explains the input you must provide to and the output you receive from the loader utility.

"Part Three: Generating and Loading Under VSE": This part of the manual covers generating and loading under the VSE operating system. It contains Chapter 7, Chapter 8, and Chapter 9:

- **Chapter 7, "Generating the Program Under VSE":** This chapter explains what you need to know to generate your NCP under the VSE operating system. It describes the generation procedure under the VSE operating system, tells you what you need to code in your job control language to generate your NCP, and describes the listings you receive from the generation procedure.

- **Chapter 8, "Examples of Job Control Language for Generation Under VSE":** This chapter contains examples of job control language for generating your NCP under the VSE operating system. It contains examples for the following types of generations:

  - A FASTRUN generation
  - An NCP/PEP generation
  - An NCP/PEP generation with user-written code
  - A dynamic reconfiguration generation.

- **Chapter 9, "Loading the Program Under VSE":** This chapter explains how to load your NCP phases into your communication controller under the VSE operating system. It describes the loader utility and explains the input you must provide to and the output you receive from the loader utility.

"Part Four. Glossary and Index": This part of the manual contains the glossary and the index.

Notes:

1. *The information in this manual pertains primarily to an NCP operating in network control mode, or in both network control mode and emulation mode through the Partitioned Emulation Programming (PEP) extension. You can find information on producing an operating Emulation Program for the IBM 3725 or 3720 Communication Controller (EP R4) in the Emulation Program for IBM Communication Controllers: Installation, Resource Definition, and Diagnosis manual, SC30-3338. You can find information on producing an operating Emulation Program for the IBM 3705 Communications Controller (EP/3705) in the Emulation Program for the IBM 3705: Generation and Utilities Guide and Reference manual, SC30-3242 (abbreviated title: EP/3705 Generation and Utilities Guide and Reference).*

2. *Throughout this manual, the following abbreviations are used to refer to the associated Advanced Communications Function (ACF) program products. Where it is necessary to distinguish between an ACF and non-ACF version of one of these products, the manual makes this distinction clear. Otherwise, always assume that an abbreviation without an "ACF/" before it is the ACF version.*

Program Product	Abbreviation
Advanced Communications Function for Network Control Program	NCP
Advanced Communications Function for Network Control Program Subset	NCP Subset
Advanced Communications Function for System Support Programs	SSP

Program Product	Abbreviation
Advanced Communications Function for Telecommunications Access Method	TCAM
Advanced Communications Function for Virtual Telecommunications Access Method	VTAM

3. *In this manual, numbers over one thousand are represented in metric style. A space is used instead of a comma to separate groups of three digits. For example, the number ten thousand, five hundred and fifty-two is written 10 552.*

Prerequisite Knowledge and Manuals

Before using this manual, you must have a thorough knowledge of Systems Network Architecture (SNA) and of the functions the NCP provides in an SNA network. You also should have a good understanding of the communication controller your NCP will reside in and the access method or access methods it will communicate with. This manual also requires a good understanding of the operating system under which you are generating and loading the NCP load module or phases. The manuals listed can help you get this prerequisite knowledge and understanding.

For All Users:

Abbreviated Title	Full Title	Order Number
SNA Concepts and Products	*Systems Network Architecture: Concepts and Products*	GC30-3072
SNA Technical Overview	*Systems Network Architecture: Technical Overview*	GC30-3073

For Users of NCP Version 4:

Abbreviated Title	Full Title	Order Number
	Network Program Products: General Information	GC30-3350
	Network Program Products: Planning	SC30-3351
	Introduction to the IBM 3725 Communication Controller (Model 1 only)	GA33-0010

Abbreviated Title	Full Title	Order Number
	Introduction to the IBM 3725 Communication Controller (Model 2 only)	GA33-0021
	IBM 3725 Communication Controller: Operator's Guide	GA33-0014
	IBM 3725 Communication Controller: Problem Determination and Extended Services	GA33-0044
	Introduction to the IBM 3720 Communication Controller	GA33-0060
	IBM 3720 Communication Controller: Operator's Guide	GA33-0065
	IBM 3720 Communication Controller: Problem Determination and Extended Services	GA33-0066

For Users of NCP Version 3 for the IBM 3725:

Abbreviated Title	Full Title	Order Number
	Introduction to the IBM 3725 Communication Controller	GA33-0010
	IBM 3725 Communication Controller: Principles of Operation	GA33-0013

For Users of NCP Version 3 for the IBM 3705:

Abbreviated Title	Full Title	Order Number
	Introduction to the IBM 3704 and 3705 Communications Controllers	GA27-3051
	IBM 3704 and 3705 Communications Controllers: Principles of Operation	GC30-3004
	Introduction to the IBM 3705-80 Communications Controller	GA27-3304
	IBM 3705-80 Communications Controller: Principles of Operation	GC30-3074

For Users of VTAM Version 3:

Abbreviated Title	Full Title	Order Number
	Network Program Products: General Information	GC30-3350
	Network Program Products: Planning	SC30-3353

For Users of VTAM Version 2 Release 2:

Abbreviated Title	Full Title	Order Number
	Network Program Products: General Information	GC27-0657
	Network Program Products: Planning	SC27-0658

For Users of VTAM Version 2 Release 1:

Abbreviated Title	Full Title	Order Number
ACF/VTAM General Information	*Advanced Communications Function for VTAM: General Information*	GC27-0608

For Users of VTAM Version 1 Release 3:

Abbreviated Title	Full Title	Order Number
ACF/VTAM General Information	*Advanced Communications Function for VTAM General Information: Introduction*	GC27-0462
ACF/VTAM General Information	*Advanced Communications Function for VTAM General Information: Concepts*	GT27-0463

For Users of TCAM Version 2 Release 4:

Abbreviated Title	Full Title	Order Number
ACF/TCAM General Information	*Advanced Communications Function for TCAM General Information: Introduction*	GC30-3057
ACF/TCAM General Information	*Advanced Communications Function for TCAM General Information: Functional Description*	GC30-3131

Corequisite Manuals

While using this manual, you will need to refer to the following manual.

Abbreviated Title	Full Title	Order Number
ACF/NCP-SSP Resource Definition Guide	*Advanced Communications Function for Network Control Program Version 3 and 4, Advanced Communications Function for System Support Programs Version 3: Resource Definition Guide*	SC30-3349
ACF/NCP-SSP Resource Definition Reference	*Advanced Communications Function for Network Control Program Versions 3 and 4; Advanced Communications Function for System Support Programs Version 3: Resource Definition Reference*	SC30-3254

You also may need to refer to the manuals listed below.

For All Users:

Abbreviated Title	Full Title	Order Number
ACF/NCP-SSP Migration	*Advanced Communications Function for Network Control Program Versions 3 and 4, Advanced Communications Function for Systems Support Program Version 3: Migration*	SC30-3252
ACF/NCP-SSP Messages and Codes	*Advanced Communications Function for Network Control Program Version 4, Advanced Communications Function for System Support Programs Version 3: Messages and Codes*	SC30-3169

Abbreviated Title	Full Title	Order Number
ACF/NCP-SSP Diagnosis Guide	*Advanced Communications Function for Network Control Program Versions 3 and 4, Advanced Communications Function for System Support Programs Version 3: Diagnosis Guide*	LY30-5591
SNA Reference Summary	*Systems Network Architecture: Reference Summary*	GA27-3136
NPA Program Description/ Operations	*Network Performance Analyzer: Program Description/Operations*	SB21-2479

For Users of NCP Version 4:

Abbreviated Title	Full Title	Order Number
ACF/NCP Reference Summary and Data Areas	*Advanced Communications Function for Network Control Program Version 4; Emulation Program: Reference Summary and Data Areas*	LY30-5570
ACF/NCP Customization	*Advanced Communications Function for Network Control Program, Version 4: Customization*	LY30-5571

For Users of NCP Version 3 for the IBM 3725:

Abbreviated Title	Full Title	Order Number
ACF/NCP Reference Summary and Data Areas	*Advanced Communications Function for Network Control Program, Version 4; Emulation Program: Reference Summary and Data Areas*	LY30-5558
ACF/NCP Customization	*Advanced Communications Function for Network Control Program, Version 3: Customization*	LY30-5559

For Users of NCP Version 3 for the IBM 3705:

Abbreviated Title	Full Title	Order Number
	Network Program Products: Samples (when available)	SC27-0659
ACF/NCP Reference Summary and Data Areas	*Advanced Communications Function for Network Control Program for the IBM 3705; Emulation Program for the IBM 3705: Reference Summary and Data Areas*	LY30-5555
ACF/NCP Customization	*Advanced Communications Function for Network Control Program for the IBM 3705: Customization*	LY30-5556
	IBM 3704 and 3705 Communications Controllers Assembler Language	GC30-3003

For Users of VTAM Version 3:

Abbreviated Title	Full Title	Order Number
ACF/VTAM Installation	*Advanced Communications Function for VTAM: Installation and Resource Definition*	SC23-0111
ACF/VTAM Operation	*Advanced Communications Function for VTAM: Operation*	SC23-0113

For Users of VTAM Version 2:

Abbreviated Title	Full Title	Order Number
ACF/VTAM Installation	*Advanced Communications Function for VTAM: Installation and Resource Definition*	SC27-0610
ACF/VTAM Operation	*Advanced Communications Function for VTAM: Operation*	SC27-0612

Contents

Part One: Generating and Loading Under MVS

Chapter 1. Generating the Program Under MVS 1-1
Understanding the Generation Procedure 1-2
Generation Steps 1-2
DASD Work Space Requirements 1-5
Performance Considerations 1-5
Controlling the Generation Procedure 1-6
Specifying Data Sets Used by NDF 1-6
Specifying Parameters for NDF 1-8
The LINECNT Parameter 1-9
The FASTRUN Parameter 1-9
The ASSEMBLY Parameter 1-9
Naming Resources 1-9
Defining Region Size 1-10
Naming Load Modules 1-11
Controlling Succeeding Generation Steps 1-11
Running a FASTRUN Generation 1-11
Running a Standard NCP/PEP Generation 1-12
Running an NCP/PEP Generation with User-Written Code or Special IBM Products 1-12
Generating User-written Code and NCP Using the NDF Standard Attachment Facility 1-13
Generating User-written Code and NCP without Using the NDF Standard Attachment Facility 1-14
Running a Dynamic Reconfiguration Generation 1-16
Understanding Listings and Error Messages 1-17
Sample Generation Report 1-18
Comments 1-20

Chapter 2. Examples of Job Control Language for Generation under MVS 2-1
Example for a FASTRUN Generation 2-2
Example for an NCP/PEP Generation with Output Written to Disk 2-3
Example for an NCP/PEP Generation with Output Written to Tape 2-6
Example for an NCP/PEP Generation with User-Written Code Using the NDF Standard Attachment Facility 2-9
Example for an NCP/PEP Generation with User-Written Code Without Using the NDF Standard Attachment Facility 2-11
Example for a Dynamic Reconfiguration Generation 2-12

Chapter 3. Loading the Program Under MVS 3-1
The Loader Utility 3-2

Host Processor and Communication Controller Requirements 3-2
Input to the Loader Utility 3-2
Output from the Loader Utility 3-3
Controlling the Loader Utility 3-4
Job Control Statements 3-4
Utility Control Statement 3-4
Examples of Job and Utility Control Statements 3-7

Part Two: Generating and Loading Under VM/SP

Chapter 4. Generating the Program Under VM/SP 4-1
Understanding the Generation Procedure 4-2
Generation Steps 4-2
DASD Work Space Requirements 4-5
Performance Considerations 4-5
Controlling the Generation Procedure 4-6
Specifying Files Used by NDF 4-6
Specifying Parameters for NDF 4-8
The LINECNT Parameter 4-9
The FASTRUN Parameter 4-9
The ASSEMBLY Parameter 4-9
Naming Resources 4-9
Defining Virtual Storage 4-10
Naming Load Modules 4-11
Controlling Succeeding Generation Steps 4-11
Running a FASTRUN Generation 4-11
Running a Standard NCP/PEP Generation 4-12
Running an NCP/PEP Generation with User-Written Code or Special IBM Products 4-12
Generating User-written Code and NCP Using the NDF Standard Attachment Facility 4-13
Generating User-written Code and NCP without Using the NDF Standard Attachment Facility 4-14
Running a Dynamic Reconfiguration Generation 4-16
Understanding Listings and Error Messages 4-17
Sample Generation Report 4-18
Comments 4-20

Chapter 5. Examples of EXECs for Generation under VM/SP 5-1
Example for a FASTRUN Generation 5-2
Example for an NCP/PEP Generation with Output Written to Disk 5-4
Example for an NCP/PEP Generation with Output Written to Tape 5-10
Example for an NCP/PEP Generation with User-Written Code Using the NDF Standard Attachment Facility 5-17
Example for an NCP/PEP Generation with User-Written Code Without Using the NDF Standard Attachment Facility 5-18
Example for a Dynamic Reconfiguration Generation 5-19

Chapter 6. Loading the Program Under VM/SP 6-1
The Loader Utility 6-2
Host Processor and Communication Controller Requirements 6-2
Input to the Loader Utility 6-3
Output from the Loader Utility 6-3

Controlling the Loader Utility 6-3
VM/SP Commands 6-3
Utility Control Statement 6-4
Examples of VM/SP Commands and Utility Control Statements 6-6

Part Three: Generating and Loading Under VSE

Chapter 7. Generating the Program Under VSE 7-1
Understanding the Generation Procedure 7-2
Generation Steps 7-2
DASD Work Space Requirements 7-4
Performance Considerations 7-4
Controlling the Generation Procedure 7-5
Specifying Files Used by NDF 7-5
Specifying Parameters for NDF 7-5
The LINECNT Parameter 7-6
The FASTRUN Parameter 7-6
Naming Resources 7-6
Defining Virtual Storage 7-7
Naming Phases 7-8
Controlling Succeeding Generation Steps 7-9
Running a FASTRUN Generation 7-10
Running a Standard NCP/PEP Generation 7-10
Running an NCP/PEP Generation with User-Written Code or Special IBM Products 7-10
Running a Dynamic Reconfiguration Generation 7-11
Understanding Listings and Error Messages 7-12
Sample Generation Report 7-12
Comments 7-14

Chapter 8. Examples of Job Control Language for Generation Under VSE 8-1
Example for a FASTRUN Generation 8-2
Example for an NCP/PEP Generation 8-3
Example for an NCP/PEP Generation With User-Written Code or Special IBM Products 8-6
Example for a Dynamic Reconfiguration Generation 8-7

Chapter 9. Loading the Program Under VSE 9-1
The Loader Utility 9-2
Host Processor and Communication Controller Requirements 9-2
Input to the Loader Utility 9-2
Output from the Loader Utility 9-3
Controlling the Loader Utility 9-4
Job Control Statements 9-4
Utility Control Statement 9-5
Examples of Job and Utility Control Statements 9-6
Link-Editing Object Code into Phases 9-8

Part Four. Glossary and Index

Glossary X-1

Figures

1-1. The Generation Procedure under MVS 1-4
1-2. Label Prefixes to Avoid 1-10
1-3. Generating User-Written Code and NCP Using the NDF Standard Attachment Facility 1-13
1-4. Generating User-Written Code and NCP without Using the NDF Standard Attachment Facility 1-15
1-5. Sample from an NDF Generation Report. 1-19
4-1. The Generation Procedure under VM/SP 4-4
4-2. Label Prefixes to Avoid 4-10
4-3. Generating User-Written Code and NCP Using the NDF Standard Attachment Facility 4-13
4-4. Generating User-Written Code and NCP without Using the NDF Standard Attachment Facility 4-15
4-5. Sample from an NDF Generation Report. 4-19
7-1. The Generation Procedure under VSE 7-3
7-2. Label Prefixes to Avoid 7-7
7-3. Determining the Number of Phases for an IBM 3705 Communications Controller 7-9
7-4. Sample from an NDF Generation Report. 7-13

1

Generating and Loading Under MVS

Chapter 1 Generating the Program Under MVS
Chapter 2 Examples of Job Control Lanquage for Generation Under MVS
Chapter 3 Loading the Program Under MVS

Part One: Generating and Loading Under MVS

Chapter 1. Generating the Program Under MVS 1-1
Understanding the Generation Procedure 1-2
Controlling the Generation Procedure 1-6
Understanding Listings and Error Messages 1-17

Chapter 2. Examples of Job Control Language for Generation under MVS 2-1
Example for a FASTRUN Generation 2-2
Example for an NCP/PEP Generation with Output Written to Disk 2-3
Example for an NCP/PEP Generation with Output Written to Tape 2-6
Example for an NCP/PEP Generation with User-Written Code Using the NDF Standard Attachment Facility 2-9
Example for an NCP/PEP Generation with User-Written Code Without Using the NDF Standard Attachment Facility 2-11
Example for a Dynamic Reconfiguration Generation 2-12

Chapter 3. Loading the Program Under MVS 3-1
The Loader Utility 3-2
Controlling the Loader Utility 3-4

Chapter 1. Generating the Program Under MVS

After you install your NCP and SSP product from the tape and after you define the NCP configuration, the next step in producing an operating NCP is to generate the program. The generation procedure can be run in any host processor; it does not have to be run in the host processor that will load the NCP. However, for the NCP Subset, the generation procedure must be run in the host processor with the same operating system of the host that will load the NCP.

This chapter contains information about generating an NCP under the MVS operating system. It consists of three parts:

- Understanding the Generation Procedure
- Controlling the Generation Procedure
- Understanding Listings and Error Messages.

SSP Version 3 includes the NCP/EP Definition Facility (NDF), a program used in generating an NCP and/or EP load module. NDF can be used in performing several different tasks. These are:

- FASTRUN validation of an NCP/PEP generation definition
- Generation of an NCP/PEP load module
- Generation of an NCP/PEP Subset load module
- Generation of an NCP/PEP load module with user-written code or special IBM products
- Generation of a text data set for dynamic reconfiguration.

SSP Version 3 Release 2 contains a set of new functions, called the NDF standard attachment facility, that allows user-written generation applications to interface with NDF during an NCP generation. The NDF standard attachment facility can help ease the task of defining resources for user-written code. For more information, see "Running an NCP/PEP Generation with User-Written Code or Special IBM Products" on page 1-12.

Before running NDF you must supply Job Control Language (JCL) to control the generation procedure. NDF does not create any JCL for you. This chapter later discusses how to control the generation procedure, and in Chapter 2, "Examples of Job Control Language for Generation under MVS" this manual supplies examples of JCL for generation.

Understanding the Generation Procedure

Generating an NCP with NDF under the MVS operating system is a two step process. For a diagram of the input NDF accepts and the output it produces under the MVS operating system, see Figure 1-1 on page 1-4.

Generation Steps

Step One: The first generation step, the NDF step, consists of four phases.

In the **first phase**, the generation validation phase, NDF does the following:

- Reads your GENDECK data set containing your generation definition
- Validates the definition statements and operands coded in the generation definition
- Generates assembler language source code for the resources coded in the generation definition
- Creates link-edit control statements that will be used later to link control block objects with preassembled NCP code objects to generate an NCP load module.

If you are using the NDF standard attachment facility in SSP V3R2 to generate user-written code and NCP V4R2 or to generate user-written code and the NCP Subset, the first phase has two additional steps. After reading the GENDECK data set, NDF does the following:

- Dynamically loads one or more user-written generation load modules
- Calls routines in the user-written generation load modules to perform generation processing and allows the routines to call NDF internal routines.

The **second and third phases** are the Table 1 and Table 2 assemblies. Each of these assemblies reads the source code specification created in the generation validation phase and generates object code for the control blocks.

In the **fourth phase**, the return code summary, NDF does the following:

- Generates a composite return code that shows the success or failure of each phase
- Creates a compact listing that gives return codes for the generation validation and table assembly phases.

Step Two: The second generation step is the link edit. During this step, the control block object code produced in the first step is linked with preassembled object modules to generate a load module. If you included user-written code, special IBM program products, or block handlers in the generation definition, the appropriate object module libraries must be available during the link edit.

Note: If you want to run a FASTRUN generation to validate your generation definition without creating control blocks, or if you want to do a generation for dynamic reconfiguration, **do not** specify in your JCL that you want the link edit step to be run.

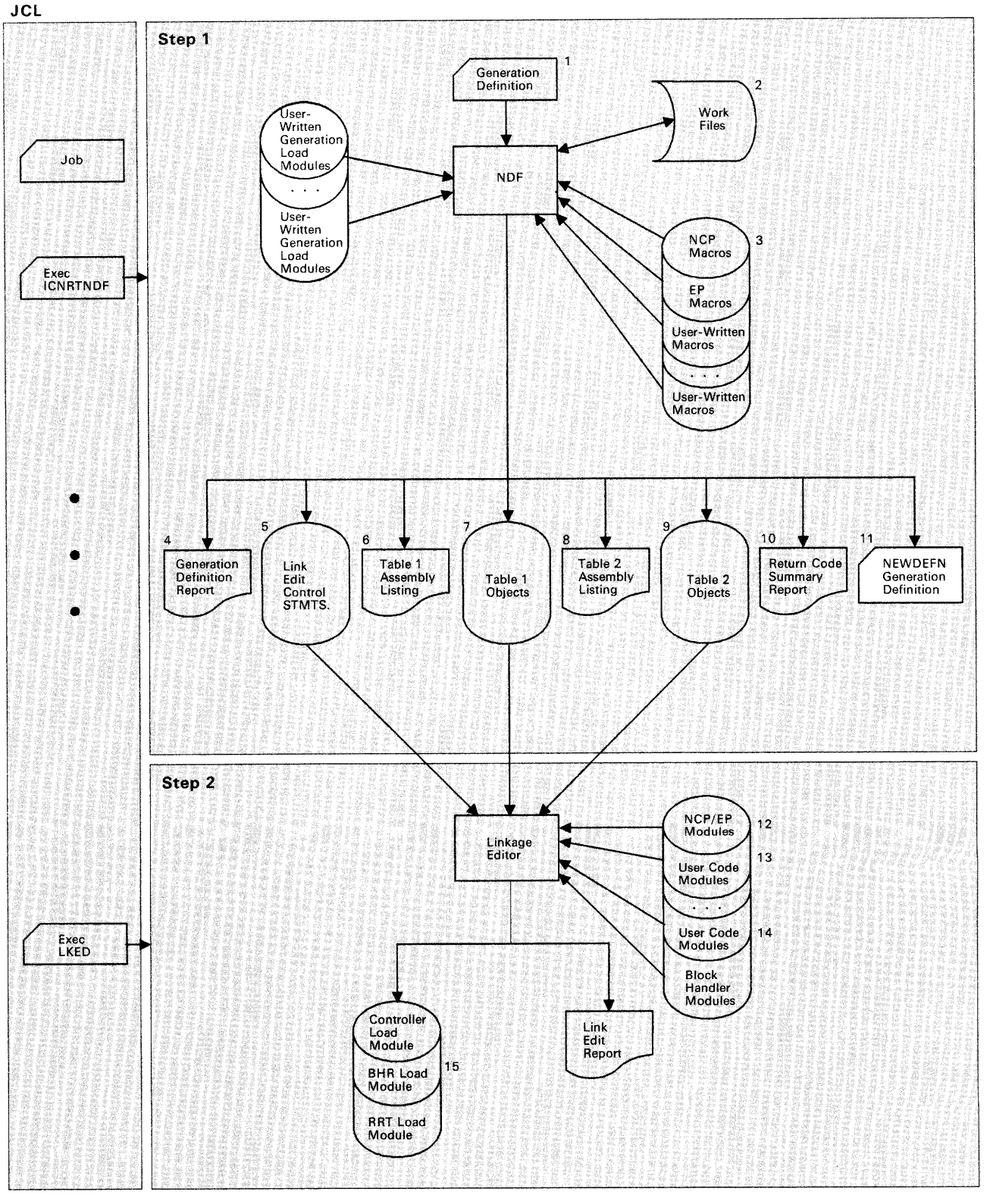


Figure 1-1. The Generation Procedure under MVS. The numbers in this figure correspond to the data sets described in "Specifying Data Sets Used by NDF" on page 1-6. The items shown in white apply to SSP V3R2 only.

DASD Work Space Requirements

NDF uses a storage manager to organize its work space during NDF generation validation. Whenever possible, storage manager data is kept in virtual storage. However, if data overflows the virtual storage region available to the storage manager, this extra data is written into a storage manager work data set.

All NDF generations require about 200K bytes of work space and an additional 48 bytes for each resource specified in the generation definition. The storage manager usually requires less than 500K bytes (K = 1024 bytes) of virtual storage. Generally, you should be able to allocate enough virtual storage to hold all the work data.

If you need additional work space, or if you are using the NDF standard attachment facility (SSP V3R2), you need to define a storage manager work data set (DBWORKFL) in your JCL. Ensure that this space allocation is not excessive, because the time required to initialize a large work data set adds a significant amount of running time for generation validation. For information about how to specify a storage manager work data set, see "Specifying Data Sets Used by NDF" on page 1-6.

Generally one cylinder of disk space allocated for a work data set should be adequate.

If you are generating NCP/Token-Ring interconnection (NTRI) resources (NCP V4R2 only), an additional 80 bytes are required for each NTRI physical line specified in the generation definition, and an additional 160 bytes are required for each NTRI logical line generated using the AUTOGEN facility.

Performance Considerations

NDF requires approximately one megabyte of real storage to achieve optimal performance. If the available real storage drops below one megabyte, paging during the generation validation phase significantly degrades performance.

In addition, a block size of at least 3 630 bytes for the Table 1 listing data set is recommended. Using even larger block sizes can noticeably improve performance. If you define an inadequate block size, the time required for the I/O operations to the data set can significantly affect elapsed time for NDF execution.

Controlling the Generation Procedure

You control the generation procedure through the job control language (JCL) that you code and through the definition statements and operands that you coded in the generation definition.

This section explains the different types of generations you can run. It discusses some of the definition statements and operands you can code in your generation definition. It also discusses the parameters you need to code and the data sets you need to specify in your JCL. For examples of JCL for generation, see page 2-1.

Specifying Data Sets Used by NDF

This section contains the data definition names (ddnames) of the data sets, which you specify in your JCL, that are used by NDF. Below are listed the ddnames and descriptions of the data sets they specify.

Note: The numbers following the ddnames correspond to the data sets pictured in Figure 1-1 on page 1-4.

ddname	Description
STEPLIB	Specifies the library containing NDF and IHR load modules. If you have any user-written generation applications that use the NDF standard attachment facility, STEPLIB must also specify the libraries containing the user-written generation load modules.
GENDECK (1)	Specifies the data set containing the definition statements for the NCP network definition. This data set must contain 80-byte fixed format records.
DBWORKFL (2)	Specifies the storage manager work data set. This temporary data set can be used to temporarily store internal data in 4K byte records during NDF generation validation. Records in the data set are accessed with the Basic Direct Access Method (BDAM). This data set is used if NDF is not able to obtain enough virtual storage to hold all of the temporary data for the generation validation phase or if you are using the NDF standard attachment facility (SSP V3R2) to generate user-written code. Ensure that this space allocation is not excessive, because the time required to initialize a large work data set adds a significant amount of running time for generation validation.
TBL1SRCE (2)	Specifies the data set that contains the Table 1 assembly source code. This data set contains the output from generation validation and serves as the input data set for the Table 1 assembly.

TBL2SRCE (2) Specifies the data set that contains the Table 2 assembly source code. This data set contains output from the generation validation phase and serves as input for the Table 2 assembly.

SYSUT1 (2) Specifies the assembler work data set. This work data set is used to temporarily store internal NDF data for the assembly of Table 1 and Table 2 objects.

SYSLIB (3) Specifies the chain of macro libraries. The libraries needed depend on the particular NDF generation. Either the 3705, 3725, or 3720 NCP macros are always needed for the table assemblies. Additional libraries may be needed for both the generation validation phase and the table assemblies if user-written code is included in the NCP.

SYSPRINT (4) Specifies the data set that contains the generation validation listing.

LNKSTMT (5) Specifies the link-edit statement data set. This data set contains the link-edit control statements produced by NDF that are used to build the NCP load module.

SYSLIN (5) Specifies the input data set that contains the link-edit control statements passed to the linkage editor from NDF.

TBL1LIST (6) Specifies the data set for the Table 1 assembly listing. This data set can be very large, and the data control block (DCB) parameters defined for it can have a significant impact on the performance of NDF.

TBL1OBJ (7) Specifies the Table 1 object data set. This data set must be a member of a partitioned data set (PDS) that is passed to the link-edit step of an NDF generation. The member name for this data set is ICNTABL1.

SYSPUNCH (7, 9) Specifies the name of the library where the control block objects have been placed by the table assemblies.

TBL2LIST (8) Specifies the Table 2 assembly listing data set.

TBL2OBJ (9) Specifies the output data set for the control block objects generated by the Table 2 assembly. This data set must be a member of the same partitioned data set (PDS) as the TBL1OBJ data set. The member name for this data set is ICNTABL2.

PRINTER (10) Specifies the data set for the return code summary report.

NEWDEFN (11) Specifies the output data set containing the new generation definition created by NDF when you are using the NDF standard attachment facility to generate user-written code or when you are generating NTRI resources. Specify this data set if you want NDF to create a new generation definition consisting of the input NCP generation definition and the NCP statements and keywords passed to NDF from any user-written generation load modules or from NTRI processing. **(SSP V3R2 only)**.

OBJxxxx (12) Specifies the library that contains the preassembled NCP object modules. The ddname and the specific library vary with the target of the generation. If you are generating an NCP V3 for the 3705, specify OBJ3705. If you are generating a full function NCP for the 3725 or 3720, specify OBJ3725. If you are generating the NCP Subset for the 3720, specify OBJ3720.

_____ (13) Specifies a library with a ddname determined by an IBM special product or by a user. This library contains preassembled object code for user-written code modules.

ULIB (14) Specifies a library used for block-handler objects or for preassembled user-written code modules.

SYSLMOD (15) Specifies the library where the controller load modules will be placed.

ASMSRCE Specifies the assembly source code that is used as input to the NDF controller assembler when the assembly option is specified. The data set must contain 80 byte fixed format records.

ASMLIST Specifies the data set for the ASMSRCE assembly listing.

ASMOBJ Specifies the data set for the ASMSRCE object data set.

Specifying Parameters for NDF

This section contains descriptions of the optional parameters for NDF that you can specify in your JCL. When specifying more than one parameter in the parameter field, you must separate the parameters with a comma.

The LINECNT Parameter

Use the LINECNT parameter to specify the number of lines on each page of the generation validation listing and the table assembly listing. The valid range for this parameter is from 10 to 99. The default value for the validation listing and for the assembly listing is 60. If you specify a value for LINECNT, this value is used in all listings.

The following is an example of the specification of LINECNT in the JCL:

```
//STEP EXEC PGM=ICNRTNDF,PARM='LINECNT=40'
```

The FASTRUN Parameter

Use the FASTRUN parameter to check for errors before running a complete generation. A FASTRUN generation checks your generation definition for syntax and definition errors without creating control blocks or link-edit control statements.

Note: Using the FASTRUN parameter is the same as coding FASTRUN=ON on the OPTIONS definition statement as the first executable statement in your generation definition.

The following is an example of the specification of FASTRUN in the JCL:

```
//STEP EXEC PGM=ICNRTNDF,PARM='FASTRUN=ON'
```

The ASSEMBLY Parameter

Use the ASSEMBLY parameter to access the NDF controller assembler to assemble table source. This assembler can be used to assemble only table source generated by NDF or macro source.

The input and output data set names used by NDF when you specify ASSEMBLY are different from those used for the table assembly, except for the assembler work data set (SYSUT1). See "Specifying Data Sets Used by NDF" on page 1-6 for the names and descriptions of these data sets.

Note: You cannot specify both the FASTRUN parameter and the ASSEMBLY parameter for the same job step.

The following is an example of the specification of ASSEMBLY in the JCL:

```
//STEP EXEC PGM=ICNRTNDF,PARM='ASSEMBLY=YES'
```

Naming Resources

Avoid using the following prefixes when naming resources, because they are used as control block identifiers and can cause duplicate labels that result in an error message from the assembler.

$	CB	DVI	LCB	NET	RH *	TRT
AAB	CBB	DVQ	LCC	NIB	RN *	TVS
ABN	CCB	ECB	LCI	NIX	RU *	UAC
ACB	CDS	ECD	LCP	NLB	R *	UCD
ACT	CER	ECL	LCS	NLX	RMB	UIB
ACU	CGP	EML	LCW	NPB	RST	UIC
AEB	CHC	EPI	LGT	NPF	RTR	ULVSGN
AST	CHV	EQB	LKB	NQB	RVT	UNA
ATB	CIE	ERB	LKC	NQE	SCB	USC
ATP	CM	ERX	LNV	NSQ	SGE	U1
AXB	COE	FCT	LPB	NVT	SGT	VAT
BC	CPI	FLB	LRC	NVX	SID	VIT
BCU	CPN	FMT	LTC	OLL	SIT	VLB
BER	CPT	FVT	LTR	OLT	SMB	VR
BGS	CRB	GCB	LTS	PAB	SMM	VST
BH	CRP	GPT	LTV	PAD	SNP	VTS
BHD	CTP	GRW	LU	PCB	SOT	VVT
BHR	CUB	GVT	LX	PIU	SST	WCB
BHS	CY	HWE	L1B	PMF	STE	WRP
BLU	CX	HWX	L4B	PSA	STQ	WU
BOQ	DAE	IB	MBF	PSB	SUT	X
BPB	DDB	ICE	MBX	PST	SVT	XDA
BST	DIA	ICW	MCT	PUV	SXB	XDB
BTT	DPT	IDD	MDR	QAB	SYS	XDH
BTU	DQB	IDE	MIB	QCB	TCB	XID
BUE	DRS	IDL	MIF	RAT	TET	
CAB	DRX	IOB	MLT	RCB	TGB	
CAI	DSP	IRN	MMV	RCQ	TH *	
CAR	DTG	LAA	MSC	RCV	TIM	
CAT	DVB	LAB	MTF	RG	TND	

* Followed by any numerical digit 0 through 9.

Figure 1-2. Label Prefixes to Avoid. The above list should not be considered all-inclusive. Avoid names that are similar to control block acronyms.

Defining Region Size

You can control virtual storage size by specifying the REGION parameter on the JOB statement or the EXEC statement for the NDF step in your JCL.

About 2 megabytes of virtual storage are required to load the largest NDF load module. Virtual storage is required to hold work data. A region of 3 megabytes should be adequate for most NDF runs.

Below is an example of the REGION specification in the JCL for 3 megabytes of storage:

```
//NDF     EXEC  PGM=ICNRTNDF,REGION=3072K
```

Naming Load Modules

Besides creating an NCP load module, NDF also produces a resource resolution table (RRT) load module and, if you have coded any block handling routines, a block-handler set resolution table (BHR) load module. These load modules contain information required by the access method.

Use the NEWNAME operand on the BUILD definition statement to designate the names for the BHR, RRT, and NCP load modules. NDF appends a "B" to the NEWNAME value to specify the name for a BHR load module and appends an "R" to the NEWNAME value to specify the name for an RRT load module.

For information about the NEWNAME operand on the BUILD definition statement, see the *ACF/NCP-SSP Resource Definition Guide*. For information on how to code this operand, see the *ACF/NCP-SSP Resource Definition Reference*.

Controlling Succeeding Generation Steps

NDF produces an overall return code for the NDF step. This return code is printed as part of the return code summary section of the NDF report and denotes the NDF phase where an error was encountered. The overall return code also is passed to the operating system as the NDF return code.

Use the overall return code to determine if succeeding job steps will be run. To do this, code a step in the JCL to run the link edit if no error occurred in the NDF step. This technique is used in the sample JCL for an NCP/PEP generation with output written to tape on page 2-6. Listed are the overall return code values and meanings. Notice that leading zeroes are suppressed.

Value	Meaning
1	input validation error
10	table 1 error
100	table 2 error
1000	printer file error

Running a FASTRUN Generation

Do a FASTRUN generation to check for errors before running a complete generation. A FASTRUN generation checks your generation definition for syntax and definition errors without creating control blocks or link-edit control statements.

To do a FASTRUN generation, code FASTRUN=ON on the OPTIONS definition statement as the first executable statement in your generation definition, or code FASTRUN=ON as a parameter in your JCL when

calling NDF. Ensure that your JCL does not call the linkage editor. If the linkage-editor step is present, an error condition will result. Also, do not define the NCP macro library, because NDF does not run table assemblies for a FASTRUN generation. However, if you include user-written code in the generation definition, define the macro library that contains user-written link-edit control statements.

For an example of the JCL for a FASTRUN generation, see page 2-2.

Running a Standard NCP/PEP Generation

To run a standard NCP/PEP generation, you supply your generation definition as input and specify the various input and output data sets in your JCL.

You can specify that input and output data sets be written to disk, or you can specify that certain data sets be written to tape.

If you detect an error while you are generating or running your NCP, you can write to tape certain listings data sets, such as the Table 1 assembly listing, the Table 2 assembly listing, and the link-edit listing.

If you are including NTRI resources in your generation definition, you must code NEWDEFN = YES on the OPTIONS definition statement as the first executable statement in your generation definition and define the NEWDEFN data set in your JCL.

For examples of JCL for running standard NCP/PEP generations, see "Example for an NCP/PEP Generation with Output Written to Disk" on page 2-3 and "Example for an NCP/PEP Generation with Output Written to Tape" on page 2-6.

Running an NCP/PEP Generation with User-Written Code or Special IBM Products

When you include user-written code or special IBM products (for example, X.25 NCP Packet Switching Interface or Network Routing Facility) in an NCP or PEP generation, you may want to modify the basic JCL.

If you are using SSP V3R2 to generate NCP V4R2 or the NCP Subset, you can generate user-written code by providing user-written generation applications. These applications use the NDF standard attachment facility to process and pass statements and keywords to NDF during generation processing. You are not required to use this method.

If you choose to generate your NCP V4R2 or NCP Subset without using the NDF standard attachment facility, or if you are generating other valid versions and releases of NCP with user-written code using SSP Version 3, you must code link-edit statements and CSECTS for your user routine. You must also identify the location of the link-edit statements by coding operands on the GENEND definition statement.

Generating User-written Code and NCP Using the NDF Standard Attachment Facility

You can run this type of generation **only** if you are using SSP V3R2 to generate NCP V4R2 or the NCP Subset. To use the NDF standard attachment facility, you must supply a user-written generation application. For information on writing user-written code and user-written generation applications, see the *NCP Customization* manual. Figure 1-3 shows how your user-written code and user-written generation load modules are included in the generation procedure.

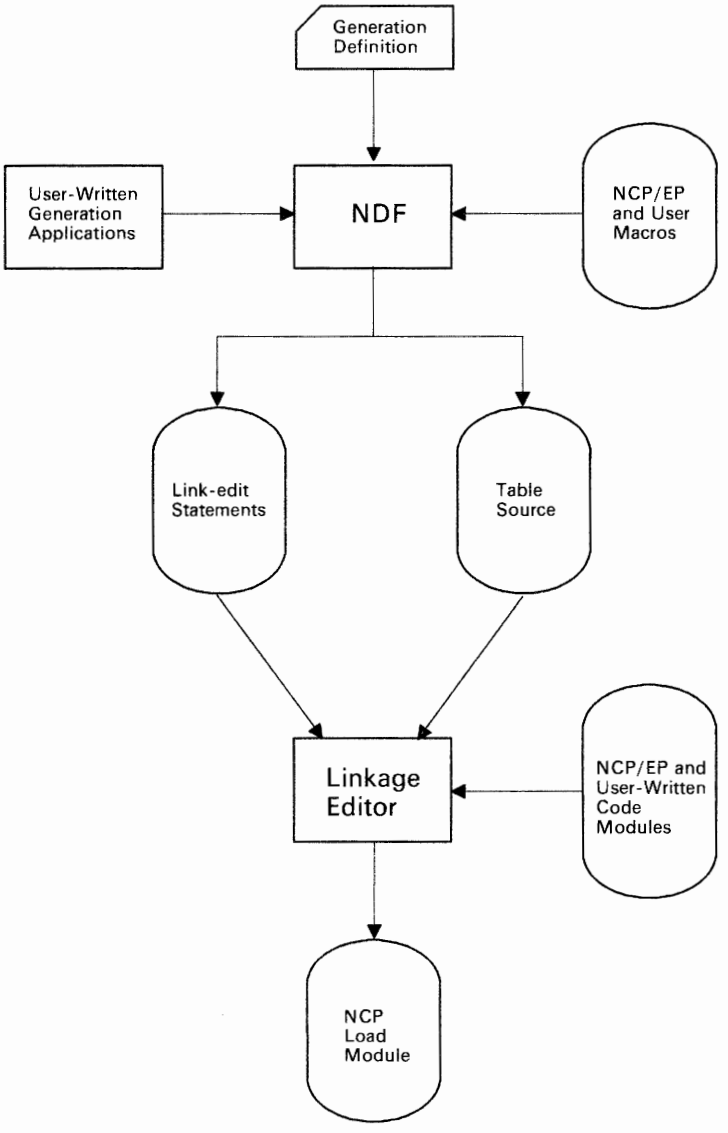


Figure 1-3. Generating User-Written Code and NCP Using the NDF Standard Attachment Facility. This figure shows how user-written generation load modules are included in an NCP/PEP generation using the NDF standard attachment facility.

Before you generate user-written code using the NDF standard attachment facility, do the following:

- Code the USERGEN operand on the OPTIONS definition statement as the first executable statement in your generation definition. The USERGEN operand specifies the names of the user-written generation load modules that will be loaded in the generation. Each application must have its own generation load module. You can specify up to 25 generation load modules.

- Code the NEWDEFN operand on the OPTIONS definition statement as the first executable statement in your generation definition if you want NDF to create a new generation definition consisting of the input NCP generation definition and the NCP statements and keywords passed to NDF from any user-written generation load modules.

- Modify the JCL for a standard NCP/PEP generation to include the ddnames for the NEWDEFN data set, the DBWORKFL data set, and the libraries for user-supplied modules.

For an example of the JCL for generating user-written code using the NDF standard attachment facility, see page 2-9.

Generating User-written Code and NCP without Using the NDF Standard Attachment Facility

You can use this type of generation for generating user-written code or special IBM products with any valid version of NCP using SSP Version 3. Before generating the NCP, code the link-edit statements for the routines and identify the location of these link-edit statements by coding certain operands on the GENEND definition statement.

Note: Besides being able to generate NCP V4R2 or the NCP Subset with user-written code using the NDF standard attachment facility as described previously, you can also generate them as described here.

Figure 1-4 on page 1-15 shows how your user-written code is included in the generation procedure.

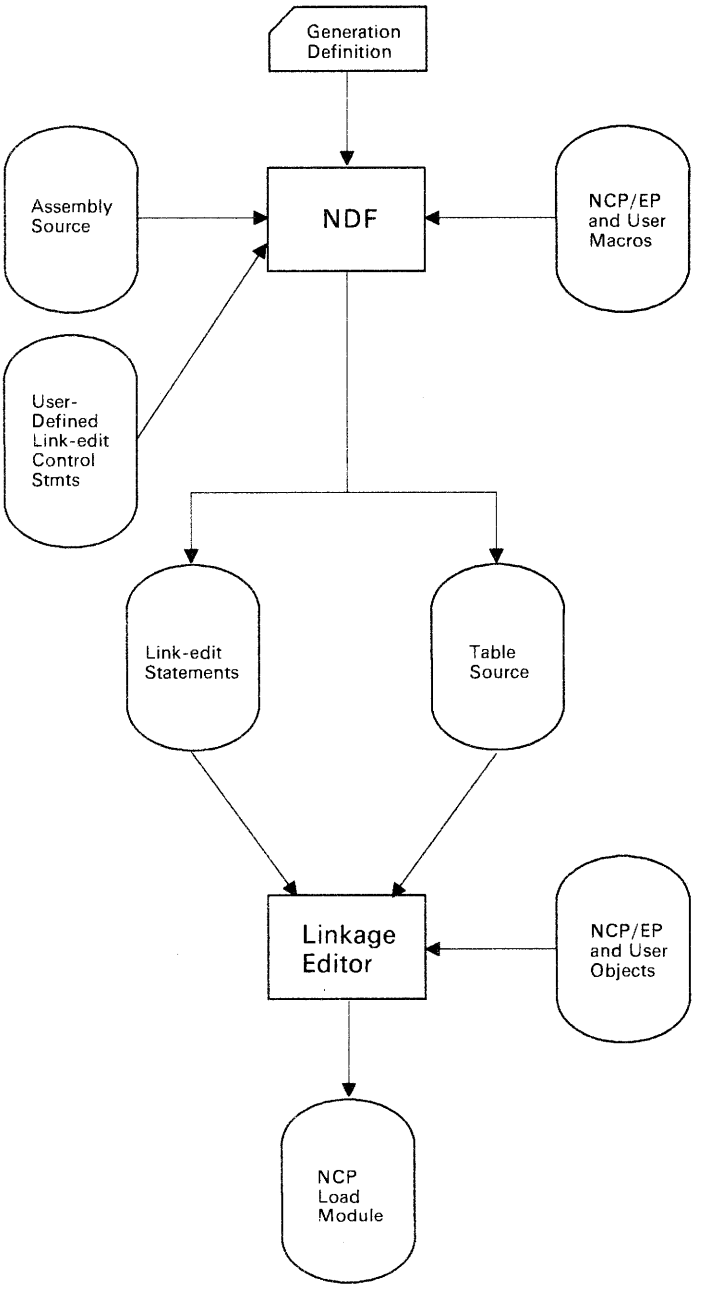


Figure 1-4. Generating User-Written Code and NCP without Using the NDF Standard Attachment Facility. This figure shows how user-written code is included in an NCP/PEP generation.

Before you generate, ensure that you:

- Assemble the user-written routines and code the link-edit control statements for the routines
- Code the appropriate operands on the GENEND definition statement for your user-written routines

- Place the members with SRCLO or SRCHI code in a macro library (SYSLIB) that is available to NDF
- Place all members that contain INCLUDE or ORDER link-edit control statements in a macro library in the NDF SYSLIB chain
- Place all macros in the NDF SYSLIB chain
- Modify the JCL to include the SYSLIB chain, and the ULIB or user object code library ddname statement.

The generation validation phase of NDF reads the link-edit control statements and writes them into the same data set as the standard NCP link-edit control statements.

For an example of JCL for generating user-written code and the NCP without using the NDF standard attachment facility, see page 2-11.

Running a Dynamic Reconfiguration Generation

Use the text data set from a dynamic reconfiguration generation to modify an NCP that is already running in a communication controller. Ensure that the original NCP was coded to allow dynamic reconfiguration. The dynamic reconfiguration generation produces a text data set that is used by one of the access methods to modify the NCP.

A dynamic reconfiguration generation requires one table assembly and no link edit.

For an example of JCL for a dynamic reconfiguration generation, see page 2-12.

Understanding Listings and Error Messages

During generation validation, NDF creates a report that contains:

- The input statements interspersed with informational and error messages
- Keywords and statements passed to NDF from user-written generation load modules using the NDF standard attachment facility (**SSP V3R2 only**)
- A resource name and network address cross reference (only in valid runs)
- An error message summary
- A return code for the generation validation step.

Also, NDF creates a listing during the return code summary phase that gives return codes for the generation definition and for each of the table assemblies.

If any errors occur in generation validation, NDF notes these errors through diagnostic messages in the report. If these messages show serious errors (severity 4 or greater), NDF ends output of control-block source and link-edit control statements, but continues to validate the input definition statements. In this case, you must correct the errors and rerun. If no serious errors occur in NDF or the table assemblies, NDF runs to completion, the link edit is run, and a load module is produced.

Note: Other procedures, such as VTAM and CRP, require as input the same definition statements coded to generate the NCP, plus several additional operands and statements specific to each procedure. The additional operands and statements are identified in the NCP/SSP Resource Definition Reference. Although you can place these additional operands and statements in the NCP generation definition either before or after you generate the NCP, it is strongly recommended that you add them before, because executing the different procedures with different inputs can create errors.

If you are using the NDF standard attachment facility (SSP V3R2) to generate user-written code with your NCP, products that use the same generation definition may require as input the source statements and/or keywords passed to NDF from the user-written generation load modules. By specifying NEWDEFN= YES on the OPTIONS definition statement in your generation definition, and by specifying the NEWDEFN data set in your JCL, you instruct NDF to create a new generation definition for use by these procedures. This new generation definition contains both the NCP source statements and the statements passed from the user-written generation load modules.

NDF does not check the accuracy of the values coded for procedures, other than the NCP generation, that appear in the NCP generation

input. It checks only the keyword part for proper spelling. In the same way, these other procedures do not check the validity of the NCP definition statements and operands. Because of this, NDF requires that the NCP generation have no errors of severity 4 or greater.

Sample Generation Report

This section contains an example of an NDF generation report. Comments about the report are cited by numbers, for example **3** , and are not a part of the actual report. You can find the comments corresponding to these numbers following the report. Ellipses (. . .) in the report indicate that parts of the report were deleted for this example.

```
1                              2                              3
ACF SSP V3R2        01/20/86 18:30:03    DEFINITION SPECIFICATION                    PAGE 1
 4
LINE #                  STATEMENT

    83  *          FULL LINE COMMENT 5
    84  PHY01      GROUP ECLTYPE=PHY          NTRI PHYSICAL GROUP 6
                     . . .
                       D                VIRTUAL=NO
                       D                TEXTTO=3.0 7
                       D                COMPACB=NO
    85  PHYLN1     LINE  ADDRESS=(80,FULL),UACB=WRONG,LOCADD=400001234567
                     . . .
                       D                MAXTSL=265
*WARNING  ICN0211  04  UACB=WRONG INVALID, INVALID NUMBER OF SUBOPERANDS, 2 EXPECTED WHEN EXLTYPE=PHYSICAL ON GROUP
8                      STATEMENT, REPLACED FOR STATEMENT KEYWORD VALIDATION
                     DELETED          UACB 9
                     ADDED            UACB(1)=X$P1AX 10
                     ADDED            UACB(2)=X$P1AR
                     G                TYPE=NCP
                     G                MAXPU=1
                     D                CLOCKNG=EXT
                     D                ATTACH=MODEM
  11                 . . .
GENERATED BY ECL
    86  X$P1SO     SERVICE
    87  ECLUPU1    PU
                     ADDED            ADDR(1)=01
                     G                PUTYPE=1
                     D                AVGPB=532
                     D                ANS=STOP
                     D                BNNSUP=()
                     D                MAXDATA=78
                     . . .
    88  ECLLU1     LU
                     ADDED            LOCADDR(1)=0
                     D                PACING(1)=1
                     D                PACING(2)=1
                     D                BATCH=NO
                     D                LUDR=NO
                   . . .
ACF SSP V3          05/09/84 13:51:30    LABEL CROSS REFERENCE                       PAGE 6

                        LABEL CROSS REFERENCE -- SORTED BY LABEL NAME 12

LABEL          LINE      SA  ELEM        LABEL          LINE      SA  ELEM        LABEL          LINE      SA  ELEM

                  . . .

G14B1          178                      T14020BB       545       OE  001B        T14022G8       915       OE  004D
G14S1          362                      T14020BC       549       OE  001C        T14022G9       916       OE  004E
                  . . .
L14022         828       OE  0044       T14020B8       533       OE  0018        T14023A7       305       OE  0009
L14023         201       OE  0001       T14020B9       537       OE  0019        T14023A8       315       OE  000A
                  . . .
ACF SSP V3          05/09/84 13:51:30    LABEL CROSS REFERENCE                       PAGE 7

                        LABEL CROSS REFERENCE -- SORTED BY NETWORK ADDRESS 13

SA  ELEM    LINE    LABEL              SA  ELEM    LINE    LABEL              SA  ELEM    LINE    LABEL

OE  0001    201     L14023             OE  0038    747     T14020F8           OE  006E    90      DRPOOLLU
OE  0002    259     B14023A            OE  0039    749     T14020F9           OE  006F    6       NCPBUILD
OE  0003    269     T14023A1           OE  003A    767     T14020G            OE  0070    6       NCPBUILD
OE  0004    275     T14023A2           OE  003B    785     T14020G1           OE  0071    6       NCPBUILD
                  . . .
ACF SSP V3R2        11/20/85 18:30:03    ERROR SUMMARY 14                            PAGE 8

TOTAL MESSAGES      INFO      WARNING      ERROR      SEVERE      FATAL
      1               0          0          1          0           0

RETURN CODE IS 8

MESSAGES APPEAR AFTER THE LINES NUMBERED:

85

REGENERATION REQUIRED
```

Figure 1-5. Sample from an NDF Generation Report.

Comments

1 This portion of the header line on the report page contains the SSP version description.

2 This portion of the header line on the report page contains the date and time of the NDF run. This is the same date and time recorded in the date/time control block in the NCP/EP load module and printed in the formatted portion of the NCP/EP dump.

3 This portion of the header line on the report page identifies the report section. It has one of the following values: "DEFINITION SPECIFICATION," "LABEL CROSS REFERENCE," or "ERROR SUMMARY."

4 The line number column contains the line numbers of the generation definition listing.

5 NDF accepts full-line assembler comments.

6 This line contains a partial-line assembler comment.

7 This part of the report contains information describing operands that are defaulted or inherited.

A code prefixed to the associated message shows operands that are defaulted or operands using values from previous definition statements. These prefixes are:

G	Keyword inherited from GROUP
L	Keyword inherited from LINE
T	Keyword inherited from TERMINAL
C	Keyword inherited from CLUSTER
P	Keyword inherited from PU
D	Keyword that has been defaulted

8 Errors are indicated by an appropriate error number followed by a severity code and error message text. A severity code of 4 or more requires corrections to the generation definition before you can generate a load module. A severity code of less than 4 informs you that NDF has taken corrective action and does not require regeneration.

9 **(Only NCP V4R2 or NCP Subset)** A DELETE message indicates operands replaced by a user-written generation application or replaced by NDF for NTRI resources.

10 **(Only NCP V4R2 or NCP Subset)** An ADDED or APPENDED message indicates operands passed to NDF by a user-written generation application or added to the generation definition by NDF for NTRI resources. An operand is ADDED if the operand was not coded or if the operand was coded but then replaced. An operand is APPENDED if the operand was coded.

11 **(Only NCP V4R2 or NCP Subset)** "GENERATED BY ECL" or "GENERATED BY usergen name" precedes statements passed to NDF by user-written generation applications using the NDF standard attachment facility, or precedes statements added to the generation definition by NDF for NTRI resources.

12 The first label cross reference is sorted by label name. All user-coded labels appear in this list. If applicable, a network address is printed. Not all labels have associated network addresses. This section is not printed when severity codes of 4 or more exist. It is included in this sample for illustrative purposes only.

13 The second label cross reference is sorted by network address. Labels with no associated network addresses are omitted. This section is not printed when severity codes of 4 or more exist.

14 The error summary section contains an error count and a list of the line numbers immediately preceding error messages. If more than one error message immediately follows a given line, the line number is printed only once. If only informational messages follow a given line, an asterisk is printed next to the line number.

Chapter 2. Examples of Job Control Language for Generation under MVS

This chapter contains examples of job control language for generating your NCP under the MVS operating system. Most of these examples can also be found on the SSP tape sent from IBM Software Distribution and are supplied here for easy reference. However, the examples of JCL for generating user-written code cannot be found on the SSP tape. Before using any examples, ensure that they match your operational environment.

This chapter includes examples of job control language for the following types of generations:

- A FASTRUN generation
- An NCP/PEP generation with output written to disk
- An NCP/PEP generation with output written to tape
- An NCP/PEP generation with user-written code using the NDF standard attachment facility
- An NCP/PEP generation with user-written code without using the NDF standard attachment facility
- A dynamic reconfiguration generation.

Note: In the examples in this chapter, the term "files" refers to data sets.

Example for a FASTRUN Generation

To do a FASTRUN generation:

- Code FASTRUN=ON on the OPTIONS definition statement as the first executable statement in your generation definition, or code FASTRUN=ON as a parameter in your JCL when calling NDF. This example uses the FASTRUN parameter coded in the JCL.
- Ensure that your JCL **does not** call the linkage editor. If the linkage-editor step is present, an error condition results.
- **Do not** define the NCP macro library, since NDF does not run table assemblies for a FASTRUN generation. However, if you include user-written code in the generation definition, define the macro library that contains user-written link-edit control statements.

Following example shows the JCL for a FASTRUN generation:

```
//MVSFAST JOB   (ACCOUNT INFO),'NAME',MSGLEVEL=(1,1)
//*
//*  COPYRIGHT=NONE
//*
//*  EXAMPLE OF A FASTRUN GENERATION.
//*
//*  THIS EXAMPLE ASSUMES THAT FASTRUN IS CODED ON THE FIRST
//*  EXECUTABLE STATEMENT IN THE GENERATION DECK.
//*
//NDF    EXEC  PGM=ICNRTNDF,REGION=3072K,PARM='FASTRUN=ON'
//* THE LIBRARY WITH NDF LOAD MODULES
//STEPLIB  DD  DSN=SYS1.SSPLIB,DISP=SHR
//* THE GENERATION DEFINITION FILE - A CARD IMAGE FILE
//GENDECK  DD  DSN=NCPSRC(SAMPLE),VOL=SER=YYYYYY,UNIT=3350,
//             DISP=SHR
//* THE REPORT FILE
//SYSPRINT DD  SYSOUT=A
//* THE SUMMARY FILE
//PRINTER  DD  SYSOUT=A
//* THE DBWORKFL IS NEEDED ONLY WHEN THERE IS NOT ENOUGH VIRTUAL MEMORY
//* TO HOLD ALL OF NDF'S WORK DATA OR IF USERGEN IS SPECIFIED.
//* //DBWORKFL DD  DSN=&&WORKF,DISP=(,DELETE),UNIT=SYSDA,
//* //             SPACE=(CYL,(1,1))
//
```

Example for an NCP/PEP Generation with Output Written to Disk

This example shows the JCL used to generate an NCP load module for the IBM 3725 Communication Controller with output data sets written to disk. For an IBM 3705 or 3720 Communication Controller:

- The JCL is slightly different.
- The NCP macro library used for the NDF job step may be different.
- The ddname for the library of preassembled NCP object modules in the link-edit step may be different.

These differences are explained in the text included in the following example.

Note: When running the link-edit step for a standard NCP/PEP generation, specify the ALIGN2 option in the JCL for the link-edit step to ensure that certain control sections within the load module are aligned on 2K boundaries. If this is not specified, the default is alignment on 4K page boundaries, which may use excessive controller storage.

Note for NTRI Users: You must code NEWDEFN=YES on the OPTIONS definition statement as the first executable statement in your generation definition and define the NEWDEFN data set in your JCL.

Following is an example for an NCP/PEP generation with output written to disk:

```
//MVSNCP JOB   (ACCOUNT INFO),'NAME',MSGLEVEL=(1,1)
//*
//*  COPYRIGHT=NONE
//*
//*  EXAMPLE OF AN NCP GENERATION WITH ALL LISTINGS WRITTEN
//*  TO DISK.
//*
//*  THIS JOB CAN ALSO BE USED FOR A GENERATION FOR IBM SPECIAL
//*  PRODUCTS WHEN THE MACROS AND OBJECT MODULES FOR THOSE PRODUCTS
//*  HAVE BEEN MERGED INTO THE APPROPRIATE NCP LIBRARIES.
//*
//*********************************************************************
//* THE FOLLOWING TABLE WILL AID YOU IN WHICH MACRO AND OBJECT        *
//* LIBRARIES CORRESPOND TO A PARTICULAR MODEL AND VERSION.  IF YOU   *
//* CHANGED YOUR LIBRARY NAMES WHEN THE PRODUCT WAS INSTALLED, YOU    *
//* MAY WANT TO UPDATE THIS TABLE.  BE SURE TO CHECK THE SYSLIB "DD"  *
//* STATEMENTS WHICH DEFINE THESE LIBRARIES.                          *
//*                                                                   *
//* NOTE:  THE HIGH LEVEL QUALIFIER FOR ALL THE LIBRARIES IS "SYS1".  *
```

This example is continued on the next page

```
//*                                                                    *
//*                            M O D E L                               *
//*                                                                    *
//*                  3705          3725          3720                 *
//*                 ---------------------------------                  *
//*      V3        |  MAC3705    |  MAC3725    |  NOT        |         *
//*  V              |  OBJ3705    |  OBJ3725    |  SUPPORTED  |         *
//*  E              |-------------------------------------------|       *
//*  R  V4          |  NOT        |  MAC3725    |  MAC3725    |         *
//*  S              |  SUPPORTED  |  OBJ3725    |  OBJ3725    |         *
//*  I              |-------------------------------------------|       *
//*  O  V4R2        |  NOT        |  MAC3725    |  MAC3725    |         *
//*  N              |  SUPPORTED  |  OBJ3725    |  OBJ3725    |         *
//*                 |-------------------------------------------|       *
//*      V4        |  NOT        |  NOT        |  MAC3725    |         *
//*      SUBSET    |  SUPPORTED  |  SUPPORTED  |  OBJ3725    |         *
//*                 ---------------------------------                  *
//*********************************************************************
//*
//* STEP TO RUN NDF
//NDF    EXEC  PGM=ICNRTNDF,REGION=3072K
//* THE LIBRARY WITH NDF AND IHR LOAD MODULES
//STEPLIB  DD  DSN=SYS1.SSPLIB,DISP=SHR
//* THE GENERATION DEFINITION FILE - A CARD IMAGE FILE
//GENDECK  DD  DSN=NCPSRC(SAMPLE),VOL=SER=YYYYYY,UNIT=3350,
//             DISP=SHR
//* THE REPORT FILE
//SYSPRINT DD  SYSOUT=A
//* THE SUMMARY FILE
//PRINTER  DD  SYSOUT=A
//* THE DBWORKFL IS NEEDED ONLY WHEN THERE IS NOT ENOUGH VIRTUAL MEMORY
//* TO HOLD ALL OF NDF'S WORK DATA OR IF USERGEN IS SPECIFIED.
//* //DBWORKFL DD  DSN=&&WORKF,DISP=(,DELETE),UNIT=SYSDA,
//* //             SPACE=(CYL,(1,1))
//* THE TABLE 1 SOURCE FILE
//TBL1SRCE DD  DSN=&&SRCE1,DISP=(,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(3,1)),DCB=BLKSIZE=3200
//* THE TABLE 1 LISTING FILE
//TBL1LIST DD  DSN=TBL1LIST,UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(CYL,(5,2))
//* THE TABLE 1 OBJECT FILE
//TBL1OBJ  DD  DSN=&&OBJ(ICNTABL1),DISP=(,PASS),UNIT=SYSDA,
//             SPACE=(CYL,(1,1,1)),DCB=BLKSIZE=3200
//* THE TABLE 2 SOURCE FILE
//TBL2SRCE DD  DSN=&&SRCE2,DISP=(,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(10,10)),DCB=BLKSIZE=3200
//* THE TABLE 2 LISTING FILE
//TBL2LIST DD  DSN=TBL2LIST,UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(CYL,(1,1))
//* THE TABLE 2 OBJECT FILE
//TBL2OBJ  DD  DSN=&&OBJ(ICNTABL2),DISP=(MOD,PASS),
//             DCB=BLKSIZE=3200,VOL=REF=*.TBL1OBJ
//* WORK FILE FOR THE TABLE ASSEMBLIES
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(10,10)),DISP=(,DELETE)
//* THE NCP MACRO LIBRARY
//SYSLIB   DD  DSN=SYS1.MAC3725,DISP=SHR
```

This example is continued on the next page

```
//* THE LINK EDIT STATEMENTS FILE
//LNKSTMT  DD  DSN=&&LNKFL,DISP=(,PASS),
//             UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=BLKSIZE=3200
//*
//* STEP TO RUN THE LINK EDIT AS LONG AS NO ERROR OCCURRED DURING NDF
//LINK   EXEC  PGM=HEWL,COND=(0,LT),REGION=3072K,
//             PARM='LIST,NCAL,MAP,ALIGN2,LET,SIZE=(3000K,20K)'
//* THE FILE OF LINK EDIT CONTROL STATEMENTS PASSED FROM STEP 1
//SYSLIN   DD  DSN=*.NDF.LNKSTMT,VOL=REF=*.NDF.LNKSTMT,
//             DISP=(OLD,DELETE)
//* THE LIBRARY OF TABLE OBJECTS PASSED FROM STEP 1
//SYSPUNCH DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//* THE LIBRARY OF PREASSEMBLED NCP OBJECT MODULES
//OBJ3725  DD  DSN=SYS1.OBJ3725,DISP=SHR
//* THE LIBRARY OF NCP LOAD MODULES
//SYSLMOD  DD  DSN=NCPLOAD,VOL=SER=YYYYYY,
//             DISP=SHR,UNIT=3350
//* THE LINK EDIT WORK FILE
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(200,20))
//SYSPRINT DD  DSN=&&LINKLIST,UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(CYL,(1,1))
//
```

Example for an NCP/PEP Generation with Output Written to Tape

If you detect an error while you are generating or running your NCP, you might want to write certain listing data sets to tape. This example shows the JCL for generating an NCP load module for an IBM 3725 Communication Controller with listing data sets from the Table 1 assembly, the Table 2 assembly, and the link edit written to tape. For an IBM 3705 or 3720 Communication Controller:

- The JCL is slightly different.
- The NCP macro library used for the NDF job step may be different.
- The ddname for the library of preassembled NCP object modules in the link-edit step may be different.

These differences are explained in the text included in the following example.

Note: When running the link-edit step for a standard NCP/PEP generation, specify the ALIGN2 option in the JCL for the link-edit step to ensure that certain control sections within the load module are aligned on 2K boundaries. If this is not specified, the default is alignment on 4K page boundaries, which may use excessive controller storage.

Note for NTRI Users: You must code NEWDEFN=YES on the OPTIONS definition statement as the first executable statement in your generation definition and define the NEWDEFN data set in your JCL.

Following is an example for an NCP/PEP generation with output written to tape:

```
//MVSTAPE JOB   (ACCOUNT INFO),'NAME',MSGLEVEL=(1,1)
//*
//*  COPYRIGHT=NONE
//*
//*  EXAMPLE OF AN NCP GENERATION WITH SOME LISTINGS WRITTEN
//*  TO TAPE.
//*
//*  THE TABLE 1, TABLE 2 AND LINK EDIT LISTINGS ARE WRITTEN TO
//*  TAPE AND PRINTED ONLY WHEN SEVERE ERRORS OCCUR.
//*
//*********************************************************************
//* THE FOLLOWING TABLE WILL AID YOU IN WHICH MACRO AND OBJECT        *
//* LIBRARIES CORRESPOND TO A PARTICULAR MODEL AND VERSION.  IF YOU   *
//* CHANGED YOUR LIBRARY NAMES WHEN THE PRODUCT WAS INSTALLED, YOU    *
//* MAY WANT TO UPDATE THIS TABLE.  BE SURE TO CHECK THE SYSLIB "DD"  *
//* STATEMENTS WHICH DEFINE THESE LIBRARIES.                          *
//*                                                                   *
//* NOTE:  THE HIGH LEVEL QUALIFIER FOR ALL THE LIBRARIES IS "SYS1".  *
```

This example is continued on the next page

```
//*                                                                     *
//*                              M O D E L                              *
//*                                                                     *
//*                    3705          3725          3720                *
//*                 ------------------------------------                *
//*       V3       |  MAC3705    |  MAC3725    |  NOT        |          *
//*   V             |  OBJ3705    |  OBJ3725    |  SUPPORTED  |          *
//*   E             |-------------------------------------------|       *
//*   R   V4        |  NOT        |  MAC3725    |  MAC3725    |          *
//*   S             |  SUPPORTED  |  OBJ3725    |  OBJ3725    |          *
//*   I             |-------------------------------------------|       *
//*   O   V4R2      |  NOT        |  MAC3725    |  MAC3725    |          *
//*   N             |  SUPPORTED  |  OBJ3725    |  OBJ3725    |          *
//*                 |-------------------------------------------|       *
//*       V4        |  NOT        |  NOT        |  MAC3725    |          *
//*       SUBSET    |  SUPPORTED  |  SUPPORTED  |  OBJ3725    |          *
//*                 ------------------------------------                *
//*********************************************************************
//*
//*  INITIALIZE TAPE
//STEP0  EXEC PGM=IEHINITT
//SYSPRINT DD SYSOUT=A
//TAPE   DD UNIT=(TP6250,1,DEFER),DCB=DEN=4,VOL=(PRIVATE,RETAIN),
//       DISP=(NEW,PASS)
//SYSIN  DD *
TAPE  INITT  SER=XXXXXX
/*
//* STEP TO RUN NDF
//NDF    EXEC  PGM=ICNRTNDF,REGION=3072K
//* THE LIBRARY WITH NDF AND IHR LOAD MODULES
//STEPLIB  DD  DSN=SYS1.SSPLIB,DISP=SHR
//* THE GENERATION DEFINITION FILE - A CARD IMAGE FILE
//GENDECK  DD  DSN=NCPSRC(SAMPLE),VOL=SER=YYYYYY,UNIT=3350,
//             DISP=SHR
//* THE REPORT FILE
//SYSPRINT DD  SYSOUT=A
//* THE SUMMARY FILE
//PRINTER  DD  SYSOUT=A
//* THE DBWORKFL IS NEEDED ONLY WHEN THERE IS NOT ENOUGH VIRTUAL MEMORY
//* TO HOLD ALL OF NDF'S WORK DATA OR IF USERGEN IS SPECIFIED.
//* //DBWORKFL DD  DSN=&&WORKF,DISP=(,DELETE),UNIT=SYSDA,
//* //            SPACE=(CYL,(1,1))
//* THE TABLE 1 SOURCE FILE
//TBL1SRCE DD  DSN=&&SRCE1,DISP=(,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(3,1)),DCB=BLKSIZE=3200
//* THE TABLE 1 LISTING FILE
//TBL1LIST DD  DSN=SAMPLE.TABLE1,UNIT=(TP6250,,DEFER),LABEL=(1,SL),
//             DISP=(OLD,PASS),
//             DCB=(DEN=4,BLKSIZE=7260,LRECL=121,RECFM=FBM),
//             VOL=(,RETAIN,,SER=XXXXXX)
//* THE TABLE 1 OBJECT FILE
//TBL1OBJ  DD  DSN=&&OBJ(ICNTABL1),DISP=(,PASS),UNIT=SYSDA,
//             SPACE=(CYL,(1,1,1)),DCB=BLKSIZE=3200
//* THE TABLE 2 SOURCE FILE
//TBL2SRCE DD  DSN=&&SRCE2,DISP=(,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(10,10)),DCB=BLKSIZE=3200
```

This example is continued on the next page

```
//* THE TABLE 2 LISTING FILE
//TBL2LIST DD  DSN=SAMPLE.TABLE2,LABEL=(2,SL),
//             VOL=REF=*.TBL1LIST,DISP=(NEW,PASS),
//             DCB=(BLKSIZE=7260,DEN=4,LRECL=121,RECFM=FBM)
//* THE TABLE 2 OBJECT FILE
//TBL2OBJ  DD  DSN=&&OBJ(ICNTABL2),DISP=(MOD,PASS),
//             DCB=BLKSIZE=3200,VOL=REF=*.TBL1OBJ
//* WORK FILE FOR THE TABLE ASSEMBLIES
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(10,10)),DISP=(,DELETE)
//* THE NCP MACRO LIBRARY
//* THE LIBRARY OF PREASSEMBLED NCP OBJECT MODULES
//SYSLIB   DD  DSN=SYS1.MAC3725,DISP=SHR
//* THE LINK EDIT STATEMENTS FILE
//LNKSTMT  DD  DSN=&&LNKFL,DISP=(,PASS),
//             UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=BLKSIZE=3200
//*
//* STEP TO PRINT OUT THE TABLE 1 LISTING AFTER AN ERROR
//ERR1   EXEC  PGM=IEBGENER,COND=(10,NE,NDF)
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=*.NDF.TBL1LIST,VOL=REF=*.NDF.TBL1LIST,
//             UNIT=TAPE,LABEL=(1,SL),DISP=(OLD,PASS)
//SYSUT2   DD  SYSOUT=A,DCB=(LRECL=121,BLKSIZE=1210,RECFM=FBM)
//SYSIN    DD  DUMMY
//*
//* STEP TO PRINT OUT THE TABLE 2 LISTING AFTER AN ERROR
//ERR2   EXEC  PGM=IEBGENER,COND=(100,NE,NDF)
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=*.NDF.TBL2LIST,VOL=REF=*.NDF.TBL2LIST,
//             UNIT=TAPE,LABEL=(2,SL),DISP=(OLD,KEEP)
//SYSUT2   DD  SYSOUT=A,DCB=(LRECL=121,BLKSIZE=1210,RECFM=FBM)
//SYSIN    DD  DUMMY
//*
//* STEP TO RUN THE LINK EDIT AS LONG AS NO ERROR OCCURRED DURING NDF
//LINK   EXEC  PGM=HEWL,COND=(0,LT),REGION=3072K,
//             PARM='LIST,NCAL,MAP,ALIGN2,LET,SIZE=(3000K,20K)'
//* THE FILE OF LINK EDIT CONTROL STATEMENTS PASSED FROM STEP 1
//SYSLIN   DD  DSN=*.NDF.LNKSTMT,VOL=REF=*.NDF.LNKSTMT,
//             DISP=(OLD,DELETE)
//* THE LIBRARY OF TABLE OBJECTS PASSED FROM STEP 1
//SYSPUNCH DD  DSN=&&OBJ,DISP=(OLD,DELETE)
//* THE LIBRARY OF PREASSEMBLED NCP OBJECT MODULES
//OBJ3725  DD  DSN=SYS1.OBJ3725,DISP=SHR
//* THE LIBRARY OF NCP LOAD MODULES
//SYSLMOD  DD  DSN=NCPLOAD,VOL=SER=YYYYYY,
//             DISP=SHR,UNIT=3350
//* THE LINK EDIT WORK FILE
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(200,20))
//SYSPRINT DD  DSN=SAMPLE.LKEDIT,LABEL=(3,SL),
//             VOL=REF=*.NDF.TBL1LIST,DISP=(NEW,PASS),
//             DCB=(BLKSIZE=1210,DEN=4,RECFM=FBA,LRECL=121)
//*
//* STEP TO PRINT THE LINK EDIT LISTING IF AN ERROR OTHER THAN
//* AN UNRESOLVED EXTERNAL REFERENCE OCCURRED
//ERR3   EXEC  PGM=IEBGENER,COND=((4,GE,LINK),(0,LT,NDF))
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=*.LINK.SYSPRINT,VOL=REF=*.LINK.SYSPRINT,
//             LABEL=(3,SL),DISP=(OLD,KEEP)
//SYSUT2   DD  SYSOUT=A,DCB=(LRECL=121,BLKSIZE=1210,RECFM=FBA)
//SYSIN    DD  DUMMY
//
```

Example for an NCP/PEP Generation with User-Written Code Using the NDF Standard Attachment Facility

This example shows the JCL for generating user-written code and NCP using the NDF standard attachment facility. For more information about running this type of generation, see page 1-12. Run this type of user-written code generation **only** if you are using SSP V3R2 to generate an NCP V4R2 or the NCP Subset.

To run an NCP/PEP generation with user-written code using the NDF standard attachment facility:

- You must supply a user-written generation application.
- You must code the USERGEN operand on the OPTIONS definition statement as the first executable statement in your generation definition.
- You can choose to code the NEWDEFN operand on the OPTIONS definition statement as the first executable statement in your generation definition if you want NDF to create a new generation definition.
- You must modify the JCL for a standard NCP/PEP generation as shown below.

Following are examples of how to code the NEWDEFN data set in the NDF step of the JCL, and how to code the ULIB data sets or the libraries for user-written code modules in the linkage-editor step of the JCL.

```
//* EXAMPLE FOR INCLUDING NEWDEFN IN THE NDF STEP IN THE JCL
//*
//*
//NEWDEFN  DD  DSN=SAMPLE.NEWDEFN,UNIT=(TP6250,,DEFER),LABEL=(1,SL),
               DISP=(OLD,PASS),
               DCB=(DEN=4,BLKSIZE=7260,LRECL=80,RECFM=FBM),
               VOL=(,RETAIN,,SER=A03338)
//*
//*


//* EXAMPLE FOR INCLUDING STEPLIB
//STEPLIB DD DSN=SYS1.SSPLIB,DISP=SHR
//        DD DSN=USER.LIB,DISP=SHR
//*
//*


//*
//*  EXAMPLE OF LIBRARIES CONCATENATED ON THE SYSLIB CHAIN
//*  IN THE NDF STEP OF A NCP GENERATION WITH USER CODE
//*
//SYSLIB   DD  DSN=V3.MAC3725,DISP=SHR,VOL=SER=DSK125,UNIT=3375
//         DD  DSN=F106MACH,DISP=SHR,VOL=SER=DSK125,UNIT=3375
//         DD  DSN=F106MACL,DISP=SHR,VOL=SER=DSK125,UNIT=3375


//*EXAMPLE FOR INCLUDING LIBRARIES FOR USER-WRITTEN CODE
//*MODULES IN THE LINK EDIT STEP
//*
//*
//* LIBRARIES FOR USER-WRITTEN CODE OBJECT MODULES
//USER1    DD  DSN=USER.OBJ1,DISP=SHR
  .        .            .
  .        .            .
  .        .            .
//USERN    DD  DSN=USER.OBJN,DISP=SHR
```

Example for an NCP/PEP Generation with User-Written Code Without Using the NDF Standard Attachment Facility

This example shows the JCL for generating user-written code and NCP without using the NDF standard attachment facility. For more information about running this type of generation, see page 1-13. Run this type of user-written code generation if you are generating any valid version and release of NCP using SSP Version 3. This includes NCP V4R2 and the NCP Subset.

Before you generate, ensure that you:

- Assemble the user-written routines and code the link-edit statements for the routines
- Code the appropriate operands on the GENEND definition statement for your user-written routines
- Place the members with SRCLO or SRCHI code in a macro library (SYSLIB) that is available to NDF
- Place all members that contain INCLUDE or ORDER link-edit control statements in a macro library in the NDF SYSLIB chain
- Place all macros in the NDF SYSLIB chain.

The generation validation phase of NDF reads the link-edit control statements and writes them into the same data set as the standard NCP link-edit control statements.

Following are examples of how to specify the SYSLIB chain and the link-edit ULIB statement. Code the SYSLIB chain in the NDF step in the JCL for a standard NCP/PEP generation, and code the ULIB statement in the link-edit step.

```
//*
//*  EXAMPLE OF LIBRARIES CONCATENATED ON THE SYSLIB CHAIN
//*  IN THE NDF STEP OF A NCP GENERATION WITH USER CODE
//*
//SYSLIB   DD  DSN=V3.MAC3725,DISP=SHR,VOL=SER=DSK125,UNIT=3375
//         DD  DSN=USERMAC1,DISP=SHR,VOL=SER=DSK125,UNIT=3375
//         DD  DSN=USERMAC2,DISP=SHR,VOL=SER=DSK125,UNIT=3375

//*    EXAMPLE OF THE ULIB DATA DEFINITION FROM THE LINK EDIT
//*    STEP OF A NCP GENERATION WITH USER CODE
//*
//*    THIS EXAMPLE DEFINES A LIBRARY OF NPSI PREASSEMBLED MODULES
//*    THE DDNAME WILL VARY FOR OTHER IBM SPECIAL PRODUCTS
//*
//ULIB     DD  DSN=NPSI,UNIT=SYSDA,DISP=SHR
//*
//*
```

Example for a Dynamic Reconfiguration Generation

Use the text data set from a dynamic reconfiguration generation to modify an NCP that is already running in a communication controller. Ensure that the original NCP was coded to allow dynamic reconfiguration. The dynamic reconfiguration generation produces a text data set that is used by one of the access methods to modify the NCP.

A dynamic reconfiguration generation requires one table assembly and no link edit.

Following is an example of JCL for a dynamic reconfiguration generation:

```
//MVSDR   JOB   (ACCOUNT INFO),'NAME',MSGLEVEL=(1,1)
//*
//*  COPYRIGHT=NONE
//*
//*  EXAMPLE OF A DYNAMIC RECONFIGURATION GENERATION.
//*
//*********************************************************************
//* THE FOLLOWING TABLE WILL AID YOU IN WHICH MACRO LIBRARY           *
//* CORRESPONDS TO A PARTICULAR MODEL AND VERSION.  IF YOU CHANGED    *
//* YOUR LIBRARY NAME WHEN THE PRODUCT WAS INSTALLED, YOU MAY WANT    *
//* TO UPDATE THIS TABLE.  BE SURE TO CHECK THE SYSLIB "DD" STATEMENT *
//* WHICH DEFINES THIS LIBRARY.                                       *
//*                                                                   *
//* NOTE:  THE HIGH LEVEL QUALIFIER FOR ALL THE LIBRARIES IS "SYS1".  *
//*                                                                   *
//*                            M O D E L                              *
//*                                                                   *
//*                 3705         3725         3720                  *
//*               ---------------------------------                   *
//*     V3      |  MAC3705   |  MAC3725   |  NOT       |             *
//*  V          |            |            |  SUPPORTED |             *
//*  E          |------------------------------------|               *
//*  R  V4      |  NOT       |  MAC3725   |  MAC3725   |             *
//*  S          |  SUPPORTED |            |            |             *
//*  I          |------------------------------------|               *
//*  O  V4R2    |  NOT       |  MAC3725   |  MAC3725   |             *
//*  N          |  SUPPORTED |            |            |             *
//*             |------------------------------------|               *
//*     V4      |  NOT       |  NOT       |  MAC3725   |             *
//*     SUBSET  |  SUPPORTED |  SUPPORTED |            |             *
//*               ---------------------------------                   *
//*********************************************************************
//*
//NDF    EXEC  PGM=ICNRTNDF,REGION=3072K
//* THE LIBRARY WITH NDF AND IHR LOAD MODULES
//STEPLIB  DD  DSN=SYS1.SSPLIB,DISP=SHR
//* THE GENERATION DEFINITION FILE - A CARD IMAGE FILE
//GENDECK  DD  DSN=DRSRC(SAMPLE),VOL=SER=YYYYYY,UNIT=3350,
//             DISP=SHR
//* THE REPORT FILE
//SYSPRINT DD  SYSOUT=A
//* THE SUMMARY FILE
//PRINTER  DD  SYSOUT=A
```

This example is continued on the next page.

```
//* THE DBWORKFL IS NEEDED ONLY WHEN THERE IS NOT ENOUGH VIRTUAL MEMORY
//* TO HOLD ALL OF NDF'S WORK DATA OR IF USERGEN IS SPECIFIED.
//* //DBWORKFL DD  DSN=&&WORKF,DISP=(,DELETE),UNIT=SYSDA,
//* //             SPACE=(CYL,(1,1))
//* THE TABLE 1 SOURCE FILE
//TBL1SRCE DD  DSN=&&SRCE1,DISP=(,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(3,1)),DCB=BLKSIZE=3200
//* THE TABLE 1 LISTING FILE
//TBL1LIST DD  DSN=&&TBL1LIST,UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(CYL,(5,2))
//* THE TABLE 1 OBJECT FILE
//TBL1OBJ  DD  DSN=&&DROUT(DRTEXT),DISP=(,PASS),UNIT=SYSDA,
//             SPACE=(CYL,(1,1,1)),DCB=BLKSIZE=3200
//* WORK FILE FOR THE TABLE ASSEMBLY
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(10,10)),DISP=(,DELETE)
//* THE NCP MACRO LIBRARY
//SYSLIB   DD  DSN=SYS1.MAC3725,DISP=SHR
//
```

Chapter 3. Loading the Program Under MVS

After generating the NCP load module, the last step in producing an operating NCP is to load the load module into the communication controller in which it is to reside. You can load your NCP into a channel-attached controller in two ways. You can use the loader utility provided by the Advanced Communications Function for System Support Programs (ACF/SSP or SSP), or you can use a loader facility provided by an access method. If you are loading your NCP into a link-attached controller, you must use the access method loader facility. This manual tells you only how to use the SSP loader utility to load a channel-attached controller. For information on how to use the access method loader facility to load both a channel-attached and link-attached controller, see the appropriate *ACF/VTAM Operation* manual and the appropriate *ACF/TCAM Base Installation Guide.*

The SSP loader utility is run as an MVS job or job step. A controller module disables all channel adapters except the one over which the load operation takes place. Upon completion of its initialization phase, the NCP enables any additional channel adapters specified as ACTIVE in the NCPCA operand of the BUILD definition statement.

Note: You must manually disable any channel adapter connected to a non-operational host before starting the load process. Successful completion of the load process is indicated to you by a write-to-operator message. A separate message is issued for each successfully loaded controller. Syntax errors or permanent I/O errors occurring during loading are indicated to you by messages sent to the message data set.

If you are loading your NCP into the IBM 3705 Communications Controller, before the loader utility loads the NCP into the controller, it can load an optional diagnostic routine called the initial test routine. If the initial test routine detects no malfunctions, the loader utility loads the NCP into the controller. If the initial test routine detects trouble, it stops, and the loader utility issues error message IFL004I indicating that fact. The loader utility then loads the remaining controllers, if any, specified in the loader job. Loading and running of the initial test routine are optional, but are recommended because the routine can detect conditions that can cause later failure of the NCP. The routine is run unless you specify its omission in the LOAD control statement.

Note: If the controller power has been turned off since the last time the initial test routine was run, run the routine again before reloading the controller. This ensures that correct parity is set in the controller storage.

The Loader Utility

This section discusses the loader utility in MVS systems. It covers the following:

- Host processor and communication controller requirements
- Input that you supply to the loader utility
- Output from the loader utility.

Host Processor and Communication Controller Requirements

The loader utility requires minimum virtual region for MVS operation.

No work data sets are required to run the loader utility.

The controller module of the loader utility can be run in any channel-attached communication controller. Before it can be loaded, the controller must:

- Have its power on
- Be identified to the operating system under which the loader utility will be run
- Be free to be allocated to the loader job step
- Not be in a program-stop condition
- Have the channel, over which the load is to take place, enabled.

Note: After the loader utility has been started, do not cancel the load job until after normal completion.

The loader utility consists of the load modules IFLOADRN, IFLLD1P1, IFLLD1P2, IFLLD2P1, and IFLLD2P2. These modules must be in the SYS1.LINKLIB data set or in a partitioned data set pointed to by a STEPLIB or JOBLIB statement.

Input to the Loader Utility

The input to the loader utility consists of two data sets. One is a DASD partitioned data set (input data set) that contains the NCP load module that you want to load into the controller. The other contains a LOAD statement specifying the name of the NCP load module and the controller into which it is to be loaded.

If you are loading your NCP into the IBM 3705 Communications Controller, the input to the loader utility consists of an additional data set. This third data set is a partitioned data set containing the initial test routine (consisting of modules IFL3705A, IFL3705B, IFL3705D, and IFL3705E), to be loaded into the controller before the NCP load module. This data set is optional. It can be omitted if you do not want the initial test routine to be run.

Output from the Loader Utility

The loader utility produces one output data set: the message data set SYSPRINT. This data set contains completion or error messages produced by the loader utility. See the *ACF/NCP-SSP Messages and Codes* manual for a description of the messages issued by the loader utility.

Controlling the Loader Utility

This section discusses the job control statements and the utility control statement that you supply to the loader utility. It also presents examples of the job and utility control statements.

Job Control Statements

The job control statements you need for calling the loader utility are:

//jobname	JOB	Starts the job.
//	EXEC	Specifies the program name, IFLOADRN, or the name of a procedure containing the job control statements.
//SYSPRINT	DD	Specifies a sequential data set. This data set can be sent to the SYSOUT device, magnetic tape volume, or direct-access volume.
//SYSUT1	DD	Specifies the DASD input data set containing the NCP load module.
//SYSUT3	DD	Only for the IBM 3705 Communications Controller. Specifies the DASD input data set containing the initial test routine load module; not required if you specify DIAG=NO in the LOAD statement.
//ccname	DD	Specifies the unit address of the controller to be loaded.
//SYSIN	DD	Specifies the data set (input stream) containing the LOAD control statement.
/*		

Utility Control Statement

Only one utility control statement is needed to call the loader utility. This is the LOAD statement. It specifies:

- Which member of the input data set contains the NCP load module to be loaded
- Which controller you will load
- Whether you will run the initial test routine, if you are loading your NCP into an IBM 3705 Communications Controller.

The following conventions are used in the description of the LOAD statement:

- Capital letters represent values you code directly, without change.
- Small letters represent parameters for which you must supply a value.
- Braces { } indicate that you must choose from the enclosed items.
- "Or" signs | indicate that you can choose between coding various operands
- An underlined value represents the default value of the operand. That is, the loader utility uses that value if you omit the operand.
- Brackets [ ] enclose operands or symbols that are either optional or conditional.

The format of the LOAD statement is:

Name	Operation	Operands
	LOAD	LOADMOD=member name, UNIT=ccname|3725=ccname|3705=ccname [,DIAG={Y6}] {Y8} {NO}

`LOADMOD=member name`

Specifies which member of the input data set indicated by SYSUT1 contains the desired NCP load module. The member must be in standard MVS load module form, without the overlay (OVLY) or scatter (SCTR) parameters.

`UNIT=ccname`

For SSP V3R2, specifies the ccname given to the data definition (DD) statement identifying the IBM 3725, 3720, or 3705 Communication Controller to be loaded.

`3725=ccname`

Specifies the ccname given to the data definition (DD) statement identifying the IBM 3725 Communication Controller to be loaded.

`3705=ccname`

Specifies the ccname given to the data definition (DD) statement identifying the IBM 3705 Communications Controller to be loaded.

```
[,DIAG={Y6}]
       {Y8}
       {NO}
```

Specifies whether the loader utility is to load the initial test routine into an IBM 3705 Communications Controller.

- **Y6** is the default and specifies that the routine is to be loaded into a controller without extended addressing. A 3705-II having 64K bytes or less of storage uses 16-bit storage addresses and, therefore, does not require extended addressing.

- **Y8** specifies that the routine is to be loaded into a controller with extended addressing. A 3705-II having more than 64K bytes of storage requires extended storage addressing.

- **NO** specifies that the initial routine is not to be loaded at all.

Examples of Job and Utility Control Statements

Example 1: (Using SSP V3R2)

Assume that you want to load an NCP load module named NCP1, residing in a data set named ONENCP, into a controller whose unit address is 030. The job and utility statements you would use to accomplish this loading into an IBM 3725 or 3720 Communication Controller are:

```
//CCLOAD        JOB         123456,SMITH,MSGLEVEL=1
//              EXEC        PGM=IFLOADRN
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSNAME=ONENCP,UNIT=3330,
//                          VOL=SER=111111,DISP=SHR
//CC030         DD          UNIT=030
//SYSIN         DD          *
              LOAD          LOADMOD=NCP1,UNIT=CC030
/*
```

The job and utility statements you would use to accomplish this loading into an IBM 3705 Communications Controller are:

```
//CCLOAD        JOB         123456,SMITH,MSGLEVEL=1
//              EXEC        PGM=IFLOADRN
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSNAME=ONENCP,UNIT=3330,
//                          VOL=SER=111111,DISP=SHR
//SYSUT3        DD          DSN=SYS1.LINKLIB,DISP=SHR
//CC030         DD          UNIT=030
//SYSIN         DD          *
              LOAD          LOADMOD=NCP1,UNIT=CC030
/*
```

Note: This example shows that the initial test routine is to be loaded and run before the NCP load module is loaded. If you did not want the initial test routine to be loaded and run, you would include DIAG=NO on the LOAD statement, and would omit the SYSUT3 DD statement.

Example 2: (Using SSP V3R2)

You can load more than one NCP into more than one controller at the same time by including a separate DD statement and LOAD statement for each controller. For example, assume that you want to load an additional NCP load module named NCP2 into a controller whose unit address is 040. Both NCP1 and NCP2 reside in a data set named TWONCPS. The job and utility statements you would use to accomplish this loading into an IBM 3725 or 3720 Communication Controller are:

```
//CCLOAD        JOB         123456,SMITH,MSGLEVEL=1
//              EXEC        PGM=IFLOADRN
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSNAME=TWONCPS,UNIT=3330,
//                          VOL=SER=111111,DISP=SHR
//CC030         DD          UNIT=030
//CC040         DD          UNIT=040
//SYSIN         DD          *
                LOAD        LOADMOD=NCP1,UNIT=CC030
                LOAD        LOADMOD=NCP2,UNIT=CC040
/*
```

The job and utility statements you would use to accomplish this loading into an IBM 3705 Communications Controller are:

```
//CCLOAD        JOB         123456,SMITH,MSGLEVEL=1
//              EXEC        PGM=IFLOADRN
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSNAME=TWONCPS,UNIT=3330,
//                          VOL=SER=111111,DISP=SHR
//CC030         DD          UNIT=030
//CC040         DD          UNIT=040
//SYSIN         DD          *
                LOAD        LOADMOD=NCP1,UNIT=CC030,DIAG=NO
                LOAD        LOADMOD=NCP2,UNIT=CC040,DIAG=NO
/*
```

Note: This example does not include loading and running of the initial test routine.

Example 3: (Using SSP V3R1)

Assume that you want to load an NCP load module named NCP1, residing in a data set named ONENCP, into a controller whose unit address is 030. The job and utility statements you would use to accomplish this loading into an IBM 3725 Communication Controller are:

```
//CCLOAD        JOB         123456,SMITH,MSGLEVEL=1
//              EXEC        PGM=IFLOADRN
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSNAME=ONENCP,UNIT=3330,
//                          VOL=SER=111111,DISP=SHR
//CC030         DD          UNIT=030
//SYSIN         DD          *
                LOAD        LOADMOD=NCP1,3725=CC030
/*
```

The job and utility statements you would use to accomplish this loading into an IBM 3705 Communications Controller are:

```
//CCLOAD        JOB         123456,SMITH,MSGLEVEL=1
//              EXEC        PGM=IFLOADRN
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSNAME=ONENCP,UNIT=3330,
//                          VOL=SER=111111,DISP=SHR
//SYSUT3        DD          DSN=SYS1.LINKLIB,DISP=SHR
//CC030         DD          UNIT=030
//SYSIN         DD          *
                LOAD        LOADMOD=NCP1,3705=CC030
/*
```

Note: This example shows that the initial test routine is to be loaded and run before the NCP load module is loaded. If you did not want the initial test routine to be loaded and run, you would include DIAG=NO on the LOAD statement, and would omit the SYSUT3 DD statement.

Example 4: (Using SSP V3R1)

You can load more than one NCP into more than one controller at the same time by including a separate DD statement and LOAD statement for each controller. For example, assume that you want to load an additional NCP load module named NCP2 into a controller whose unit address is 040. Both NCP1 and NCP2 reside in a data set named TWONCPS. The job and utility statements you would use to accomplish this loading into an IBM 3725 Communication Controller are:

```
//CCLOAD        JOB         123456,SMITH,MSGLEVEL=1
//              EXEC        PGM=IFLOADRN
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSNAME=TWONCPS,UNIT=3330,
//                          VOL=SER=111111,DISP=SHR
//CC030         DD          UNIT=030
//CC040         DD          UNIT=040
//SYSIN         DD          *
                LOAD        LOADMOD=NCP1,3725=CC030
                LOAD        LOADMOD=NCP2,3725=CC040
/*
```

The job and utility statements you would use to accomplish this loading into an IBM 3705 Communications Controller are:

```
//CCLOAD        JOB         123456,SMITH,MSGLEVEL=1
//              EXEC        PGM=IFLOADRN
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSNAME=TWONCPS,UNIT=3330,
//                          VOL=SER=111111,DISP=SHR
//CC030         DD          UNIT=030
//CC040         DD          UNIT=040
//SYSIN         DD          *
                LOAD        LOADMOD=NCP1,3705=CC030,DIAG=NO
                LOAD        LOADMOD=NCP2,3705=CC040,DIAG=NO
/*
```

Note: This example does not include loading and running of the initial test routine.

2

Generating and Loading Under VM/SP

Chapter 4 Generating the Program Under VM/SP
Chapter 5 Examples of EXECs for Generation Under VM/SP
Chapter 6 Loading the Program Under VM/SP

Part Two: Generating and Loading Under VM/SP

Chapter 4. Generating the Program Under VM/SP 4-1
Understanding the Generation Procedure 4-2
Controlling the Generation Procedure 4-6
Understanding Listings and Error Messages 4-17

Chapter 5. Examples of EXECs for Generation under VM/SP 5-1
Example for a FASTRUN Generation 5-2
Example for an NCP/PEP Generation with Output Written to Disk 5-4
Example for an NCP/PEP Generation with Output Written to Tape 5-10
Example for an NCP/PEP Generation with User-Written Code Using the NDF Standard Attachment Facility 5-17
Example for an NCP/PEP Generation with User-Written Code Without Using the NDF Standard Attachment Facility 5-18
Example for a Dynamic Reconfiguration Generation 5-19

Chapter 6. Loading the Program Under VM/SP 6-1
The Loader Utility 6-2
Controlling the Loader Utility 6-3

Chapter 4. Generating the Program Under VM/SP

After you install your NCP and SSP product from the tape and after you define the NCP configuration, the next step in producing an operating NCP is to generate the program. The generation procedure can be run in any host processor; it does not have to be run in the host processor that will load the NCP. However, for the NCP Subset, the generation procedure must be run in the host processor with the same operating system of the host that will load the NCP.

This chapter contains information about generating an NCP under the VM/SP operating system: It consists of three parts:

- Understanding the Generation Procedure
- Controlling the Generation Procedure
- Understanding Listings and Error Messages.

SSP Version 3 includes the NCP/EP Definition Facility (NDF), a program used in generating an NCP and/or EP load module. NDF can be used in performing several different tasks. These are:

- FASTRUN validation of an NCP/PEP generation definition
- Generation of an NCP/PEP load module
- Generation of an NCP/PEP Subset load module
- Generation of an NCP/PEP load module with user-written code or special IBM products
- Generation of a text file for dynamic reconfiguration.

SSP Version 3 Release 2 contains a set of new functions, called the NDF standard attachment facility, that allows user-written generation applications to interface with NDF during an NCP generation. The NDF standard attachment facility can help ease the task of defining resources for user-written code. For more information, see "Running an NCP/PEP Generation with User-Written Code or Special IBM Products" on page 4-12.

Before running NDF you must supply EXECs to control the generation procedure. NDF does not create any EXECs for you. This chapter later discusses how to control the generation procedure, and in Chapter 5, "Examples of EXECs for Generation under VM/SP" this manual supplies examples of EXECs for generation.

Understanding the Generation Procedure

Generating an NCP with NDF under the VM/SP operating system is a two step process. For a diagram of the input NDF accepts and the output it produces under the VM/SP operating system, see Figure 4-1 on page 4-4.

Generation Steps

Step One: The first generation step, the NDF step, consists of four phases.

In the **first phase**, the generation validation phase, NDF does the following:

- Reads your GENDECK file containing your generation definition
- Validates the definition statements and operands coded in the generation definition
- Generates assembler language source code for the resources coded in the generation definition
- Creates link-edit control statements that are will be used later to link control block objects with preassembled NCP code objects to generate an NCP load module.

If you are using the NDF standard attachment facility in SSP V3R2 to generate user-written code and NCP V4R2 or to generate user-written code and the NCP Subset, the first phase has two additional steps. After reading the GENDECK file, NDF does the following:

- Dynamically loads one or more user-written generation load modules
- Calls routines in the user-written generation load modules to perform generation processing and allows the routines to call NDF internal routines.

The **second and third phases** are the Table 1 and Table 2 assemblies. Each of these assemblies reads the source code specification created in the generation validation phase and generates object code for the control blocks.

In the **fourth phase**, the return code summary, NDF does the following:

- Generates a composite return code that shows the success or failure of each phase
- Creates a compact listing that gives return codes for the generation validation and table assembly phases.

Step Two: The second generation step is the link edit. During this step, the control block object code produced in the first step is linked with preassembled object modules to generate a load module. If you included user-written code, special IBM program products, or block handlers in the generation definition, the appropriate object module libraries must be available during the link edit.

Note: If you want to run a FASTRUN generation to validate your generation definition without creating control blocks, or if you want to do a generation for dynamic reconfiguration, **do not** specify in your EXEC that you want the link edit step to be run.

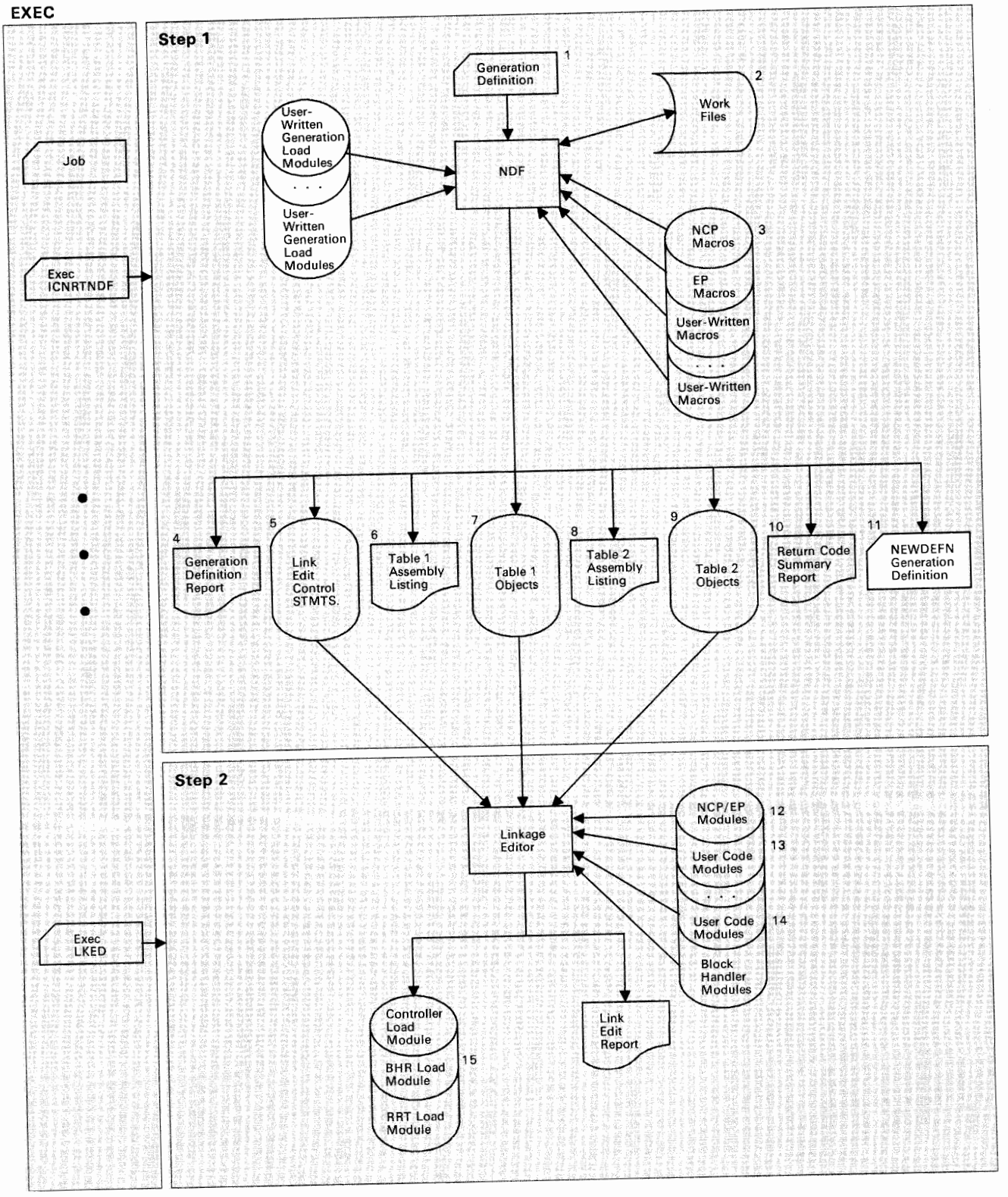


Figure 4-1. The Generation Procedure under VM/SP. The numbers in this figure correspond to the files described in "Specifying Files Used by NDF" on page 4-6. The items shown in white apply to SSP V3R2 only.

DASD Work Space Requirements

NDF uses a storage manager to organize its work space during NDF generation validation. Whenever possible, storage manager data is kept in virtual storage. However, if data overflows the virtual storage region available to the storage manager, this extra data is written into a storage manager work file.

All NDF generations require about 200K bytes of work space and an additional 48 bytes for each resource specified in the generation definition. The storage manager usually requires less than 500K bytes (K = 1024 bytes) of virtual storage. Generally, you should be able to allocate enough virtual storage to hold all the work data.

If you need additional work space, or if you are using the NDF standard attachment facility (SSP V3R2), you need to define a storage manager work file (DBWORKFL) in your EXEC. Ensure that this space allocation is not excessive, because the time required to initialize a large work file adds a significant amount of running time for generation validation. For information about how to specify a storage manager work file, see "Specifying Files Used by NDF" on page 4-6.

Generally one cylinder of disk space allocated for a work file should be adequate.

If you are generating NCP/Token-Ring interconnection (NTRI) resources (NCP V4R2 only), an additional 80 bytes are required for each NTRI physical line specified in the generation definition, and an additional 160 bytes are required for each NTRI logical line generated using the AUTOGEN facility.

Performance Considerations

NDF requires approximately one megabyte of real storage to achieve optimal performance. If the available real storage drops below one megabyte, paging during the generation validation phase significantly degrades performance.

In addition, a block size of at least 3 630 bytes for the Table 1 listing file is recommended. Using even larger block sizes may noticeably improve performance. If you define an inadequate block size, the time required for the I/O operations to the file can significantly affect elapsed time for NDF execution.

Controlling the Generation Procedure

You control the generation procedure through the EXEC that you code and through the definition statements and operands that you coded in the generation definition.

This section explains the different types of generations you can run. It discusses some of the definition statements and operands you can code in your generation definition. It also discusses the parameters you need to code and the files you need to specify in your EXEC. For examples of EXECs for generating, see page 5-1.

Specifying Files Used by NDF

This section contains the data definition names (ddnames) of the files, which you specify in your EXEC, that are used by NDF. Below are listed the ddnames and descriptions of the files they specify.

Note: The numbers following the ddnames correspond to the files pictured in Figure 4-1 on page 4-4.

ddname	Description
STEPLIB	Specifies the library containing NDF and IHR load modules. If you have any user-written generation applications that use the NDF standard attachment facility, STEPLIB must also specify the libraries containing the user-written generation load modules.
GENDECK (1)	Specifies the file containing the definition statements for the NCP network definition. This file must contain 80-byte fixed format records.
DBWORKFL (2)	Specifies the storage manager work file. This temporary file can be used to temporarily store internal data in 4K byte records during NDF generation validation. Records in the file are accessed with the Basic Direct Access Method (BDAM). This file is used if NDF is not able to obtain enough virtual storage to hold all of the temporary data for the generation validation phase or if you are using the NDF standard attachment facility (SSP V3R2) to generate user-written code. Ensure that this space allocation is not excessive, because the time required to initialize a large work file adds a significant amount of running time for generation validation.
TBL1SRCE (2)	Specifies the file that contains the Table 1 assembly source code. This file contains the output from generation validation and serves as the input file for the Table 1 assembly.

TBL2SRCE (2) Specifies the file that contains the Table 2 assembly source code. This file contains output from the generation validation phase and serves as input for the Table 2 assembly.

SYSUT1 (2) Specifies the assembler work file. This work file is used to temporarily store internal NDF data for the assembly of Table 1 and Table 2 objects.

SYSLIB (3) Specifies the chain of macro libraries. The libraries needed depend on the particular NDF generation. Either the 3705, 3725, or 3720 NCP macros are always needed for the table assemblies. Additional libraries may be needed for both the generation validation phase and the table assemblies if user-written code is included in the NCP.

SYSPRINT (4) Specifies the file that contains the generation validation listing.

LNKSTMT (5) Specifies the link-edit statement file. This file contains the link-edit control statements produced by NDF that are used to build the NCP load module.

SYSLIN (5) Specifies the input file that contains the link-edit control statements passed to the linkage editor from NDF.

TBL1LIST (6) Specifies the file for the Table 1 assembly listing. This file can be very large, and the data control block (DCB) parameters defined for it can have a significant impact on the performance of NDF.

TBL1OBJ (7) Specifies the Table 1 object file. This file must be a member of a partitioned data set (PDS) that is passed to the link-edit step of an NDF generation. The member name for this file is ICNTABL1.

SYSPUNCH (7, 9) Specifies the name of the library where the control block objects have been placed by the table assemblies.

TBL2LIST (8) Specifies the Table 2 assembly listing file.

TBL2OBJ (9) Specifies the output file for the control block objects generated by the Table 2 assembly. This file must be a member of the same partitioned data set (PDS) as the TBL1OBJ file. The member name for this file is ICNTABL2.

PRINTER (10) Specifies the file for the return code summary report.

NEWDEFN (11) Specifies the output file containing the new generation definition created by NDF when you are using the NDF standard attachment facility to generate user-written code or when you are generating NTRI resources. Specify this file if you want NDF to create a new generation definition consisting of the input NCP generation definition and the NCP statements and keywords passed to NDF from any user-written generation load modules or from NTRI processing. **(SSP V3R2 only)**.

OBJxxxx (12) Specifies the library that contains the preassembled NCP object modules. The ddname and the specific library vary with the target of the generation. If you are generating an NCP V3 for the 3705, specify OBJ3705. If you are generating a full function NCP for the 3725 or 3720, specify OBJ3725. If you are generating the NCP Subset for the 3720, specify OBJ3720.

____ (13) Specifies a library with a ddname determined by an IBM special product or by a user. This library contains preassembled object code for user-written code modules.

ULIB (14) Specifies a library used for block-handler objects or for preassembled user-written code modules.

SYSLMOD (15) Specifies the library where the controller load modules will be placed.

ASMSRCE Specifies the assembly source code that is used as input to the NDF controller assembler when the assembly option is specified. The file must contain 80 byte fixed format records.

ASMLIST Specifies the file for the ASMSRCE assembly listing.

ASMOBJ Specifies the file for the ASMSRCE object file.

Specifying Parameters for NDF

This section contains descriptions of the optional parameters for NDF that you can specify in your EXEC. When specifying more than one parameter in the parameter field, you must separate the parameters with one or more spaces. The right parenthesis closing the parameter list is not required.

The LINECNT Parameter

Use the LINECNT parameter to specify the number of lines on each page of the generation validation listing and the table assembly listing. The valid range for this parameter is from 10 to 99. The default value for the validation listing and for the assembly listing is 60. If you specify a value for LINECNT, this value is used in all listings.

The following is an example of the specification of LINECNT in the EXEC:

```
ICNRTNDF (LINECNT(40))
```

The FASTRUN Parameter

Use the FASTRUN parameter to check for errors before running a complete generation. A FASTRUN generation checks your generation definition for syntax and definition errors without creating control blocks or link-edit control statements.

Note: Using the FASTRUN parameter is the same as coding FASTRUN=ON on the OPTIONS definition statement as the first executable statement in your generation definition.

The following is an example of the specification of FASTRUN in the EXEC:

```
ICNRTNDF (FASTRUN(ON))
```

The ASSEMBLY Parameter

Use the ASSEMBLY parameter to access the NDF controller assembler to assemble table source. This assembler can be used to assemble only table source generated by NDF or macro source.

The input and output file names used by NDF when you specify ASSEMBLY are different from those used for the table assembly, except for the assembler work file (SYSUT1). See "Specifying Files Used by NDF" on page 4-6 for the names and descriptions of these files.

Note: You cannot specify both the FASTRUN parameter and the ASSEMBLY parameter for the same job step.

The following is an example of the specification of ASSEMBLY in the EXEC:

```
ICNRTNDF (ASSEMBLY(YES))
```

Naming Resources

Avoid using the following prefixes when naming resources, because they are used as control block identifiers and can cause duplicate labels that result in an error message from the assembler.

$	CB	DVI	LCB	NET	RH *	TRT
AAB	CBB	DVQ	LCC	NIB	RN *	TVS
ABN	CCB	ECB	LCI	NIX	RU *	UAC
ACB	CDS	ECD	LCP	NLB	R *	UCD
ACT	CER	ECL	LCS	NLX	RMB	UIB
ACU	CGP	EML	LCW	NPB	RST	UIC
AEB	CHC	EPI	LGT	NPF	RTR	ULVSGN
AST	CHV	EQB	LKB	NQB	RVT	UNA
ATB	CIE	ERB	LKC	NQE	SCB	USC
ATP	CM	ERX	LNV	NSQ	SGE	U1
AXB	COE	FCT	LPB	NVT	SGT	VAT
BC	CPI	FLB	LRC	NVX	SID	VIT
BCU	CPN	FMT	LTC	OLL	SIT	VLB
BER	CPT	FVT	LTR	OLT	SMB	VR
BGS	CRB	GCB	LTS	PAB	SMM	VST
BH	CRP	GPT	LTV	PAD	SNP	VTS
BHD	CTP	GRW	LU	PCB	SOT	VVT
BHR	CUB	GVT	LX	PIU	SST	WCB
BHS	CY	HWE	L1B	PMF	STE	WRP
BLU	CX	HWX	L4B	PSA	STQ	WU
BOQ	DAE	IB	MBF	PSB	SUT	X
BPB	DDB	ICE	MBX	PST	SVT	XDA
BST	DIA	ICW	MCT	PUV	SXB	XDB
BTT	DPT	IDD	MDR	QAB	SYS	XDH
BTU	DQB	IDE	MIB	QCB	TCB	XID
BUE	DRS	IDL	MIF	RAT	TET	
CAB	DRX	IOB	MLT	RCB	TGB	
CAI	DSP	IRN	MMV	RCQ	TH *	
CAR	DTG	LAA	MSC	RCV	TIM	
CAT	DVB	LAB	MTF	RG	TND	

* Followed by any numerical digit 0 through 9.

Figure 4-2. Label Prefixes to Avoid. The above list should not be considered all-inclusive. Avoid names that are similar to control block acronyms.

Defining Virtual Storage

You can control virtual storage size by using the CP DEFINE STORAGE command in your EXEC. The storage specified must include space for CMS and that space required by NDF.

About 2 megabytes of virtual storage are required to load the largest NDF load module. Virtual storage is required to hold work data. A region of 3 megabytes should be adequate for most NDF runs.

Below is an example of the CP DEFINE STORAGE command for 3 megabytes of storage:

```
CP DEFINE STORAGE AS 3072K
```

Naming Load Modules

Besides creating an NCP load module, NDF also produces a resource resolution table (RRT) load module and, if you have coded any block handling routines, a block-handler set resolution table (BHR) load module. These load modules contain information required by the access method.

Use the NEWNAME operand on the BUILD definition statement to designate the names for the BHR, RRT, and NCP load modules. NDF appends a "B" to the NEWNAME value to specify the name for a BHR load module and appends an "R" to the NEWNAME value to specify the name for an RRT load module.

For information about the NEWNAME operand on the BUILD definition statement, see the *ACF/NCP-SSP Resource Definition Guide*. For information on how to code this operand, see the *ACF/NCP-SSP Resource Definition Reference*.

Controlling Succeeding Generation Steps

NDF produces an overall return code for the NDF step. This return code is printed as part of the return code summary section of the NDF report and denotes the NDF phase where an error was encountered. The overall return code also is passed to the operating system as the NDF return code.

Use the overall return code to determine if succeeding steps will be run. This technique is used in the sample EXEC for an NCP/PEP generation with output written to tape on page 5-10. Listed are the overall return code values and meanings. Notice that leading zeroes are suppressed.

Value	Meaning
1	input validation error
10	table 1 error
100	table 2 error
1000	printer file error
10000	error freeing auxiliary directory (CMS)

Running a FASTRUN Generation

Do a FASTRUN generation to check for errors before running a complete generation. A FASTRUN generation checks your generation definition for syntax and definition errors without creating control blocks or link-edit control statements.

To do a FASTRUN generation, code FASTRUN=ON on the OPTIONS definition statement as the first executable statement in your generation definition, or code FASTRUN=ON as a parameter in your EXEC when

calling NDF. Ensure that your EXEC does not call the linkage editor. If the linkage-editor step is present, an error condition will result. Also, do not define the NCP macro library, since NDF does not run table assemblies for a FASTRUN generation. However, if you include user-written code in the generation definition, define the macro library that contains user-written link-edit control statements.

For an example of the EXEC for a FASTRUN generation, see page 5-2.

Running a Standard NCP/PEP Generation

To run a standard NCP/PEP generation, you supply your generation definition as input and specify the various input and output files in your EXEC.

You can specify that input and output files be written to disk, or you can specify that certain files be written to tape.

If you detect an error while you are generating or running your NCP, you can write to tape certain listings files, such as the Table 1 assembly listing, the Table 2 assembly listing, and the link-edit listing.

If you are including NTRI resources in your generation definition, you must code NEWDEFN=YES on the OPTIONS definition statement as the first executable statement in your generation definition and define the NEWDEFN file in your EXEC.

For examples of EXECs for running standard NCP/PEP generations, see "Example for an NCP/PEP Generation with Output Written to Disk" on page 5-4 and "Example for an NCP/PEP Generation with Output Written to Tape" on page 5-10.

Running an NCP/PEP Generation with User-Written Code or Special IBM Products

When you include user-written code or special IBM products (for example, X.25 NCP Packet Switching Interface or Network Routing Facility) in an NCP or PEP generation, you may need to modify the basic EXEC.

If you are using SSP V3R2 to generate NCP V4R2 or the NCP Subset, you can generate user-written code by providing user-written generation applications. These applications use the NDF standard attachment facility to process and pass statements and keywords to NDF during generation processing. You are not required to use this method.

If you choose to generate your NCP V4R2 or NCP Subset without using the NDF standard attachment facility, or if you are generating other valid versions and releases of NCP with user-written code using SSP Version 3, you must code link-edit statements and CSECTS for your user routine. You must also identify the location of the link-edit statements by coding operands on the GENEND definition statement.

Generating User-written Code and NCP Using the NDF Standard Attachment Facility

You can run this type of generation **only** if you are using SSP V3R2 to generate NCP V4R2 or the NCP Subset. To use the NDF standard attachment facility, you must supply a user-written generation application. For information on writing user-written code and user-written generation applications, see the *NCP Customization* manual. Figure 4-3 shows how your user-written code and user-written generation load modules are included in the generation procedure.

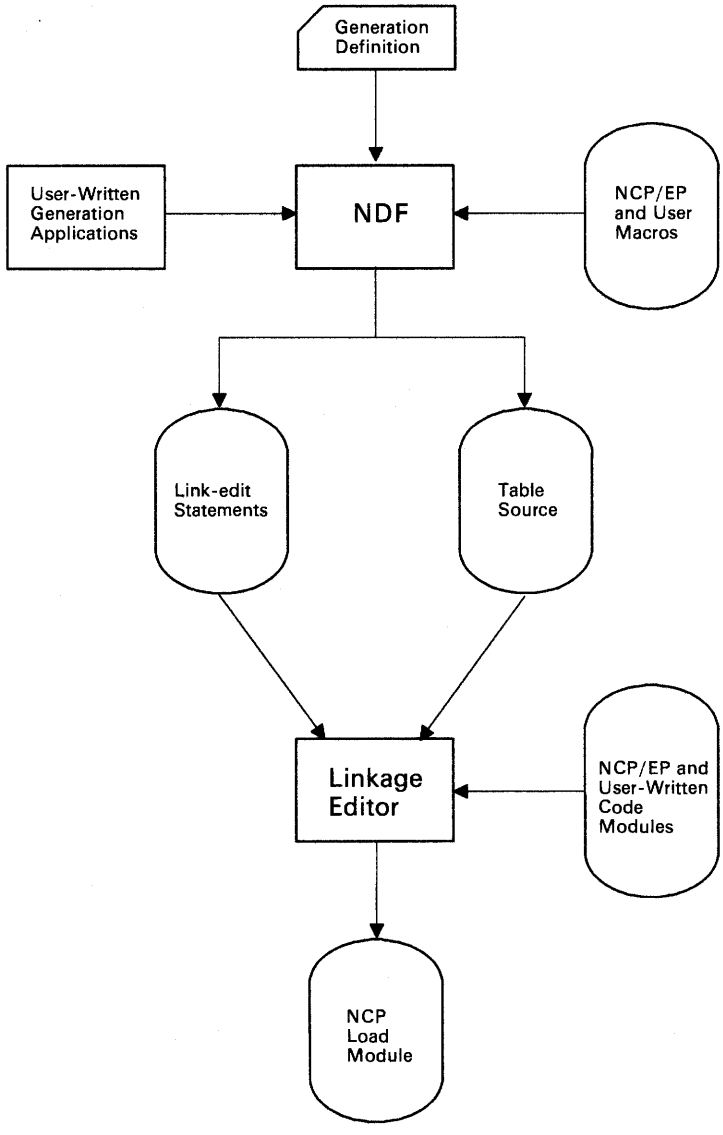


Figure 4-3. Generating User-Written Code and NCP Using the NDF Standard Attachment Facility. This figure shows how user-written generation load modules are included in an NCP/PEP generation using the NDF standard attachment facility.

Before you generate user-written code using the NDF standard attachment facility, do the following:

- Code the USERGEN operand on the OPTIONS definition statement as the first executable statement in your generation definition. The USERGEN operand specifies the names of the user-written generation load modules that will be loaded in the generation. Each application must have its own generation load module. You can specify up to 25 generation load modules.

- Code the NEWDEFN operand on the OPTIONS definition statement as the first executable statement in your generation definition if you want NDF to create a new generation definition consisting of the input NCP generation definition and the NCP statements and keywords passed to NDF from any user-written generation load modules.

- Modify the EXEC for a standard NCP/PEP generation to include the ddnames for the NEWDEFN file, the DBWORKFL file, and the libraries for user-supplied modules.

For an example of the EXEC for generating user-written code using the NDF standard attachment facility, see page 5-17.

Generating User-written Code and NCP without Using the NDF Standard Attachment Facility

You can use this type of generation for generating user-written code or special IBM products with any valid version of NCP using SSP Version 3. Before generating the NCP, code the link-edit statements for the routines and identify the location of these link-edit statements by coding certain operands on the GENEND definition statement.

Note: Besides being able to generate user-written code and NCP V4R2 or the NCP Subset using the NDF standard attachment facility as described previously, you can also generate them as described here.

Figure 4-4 on page 4-15 shows how your user-written code is included in the generation procedure.

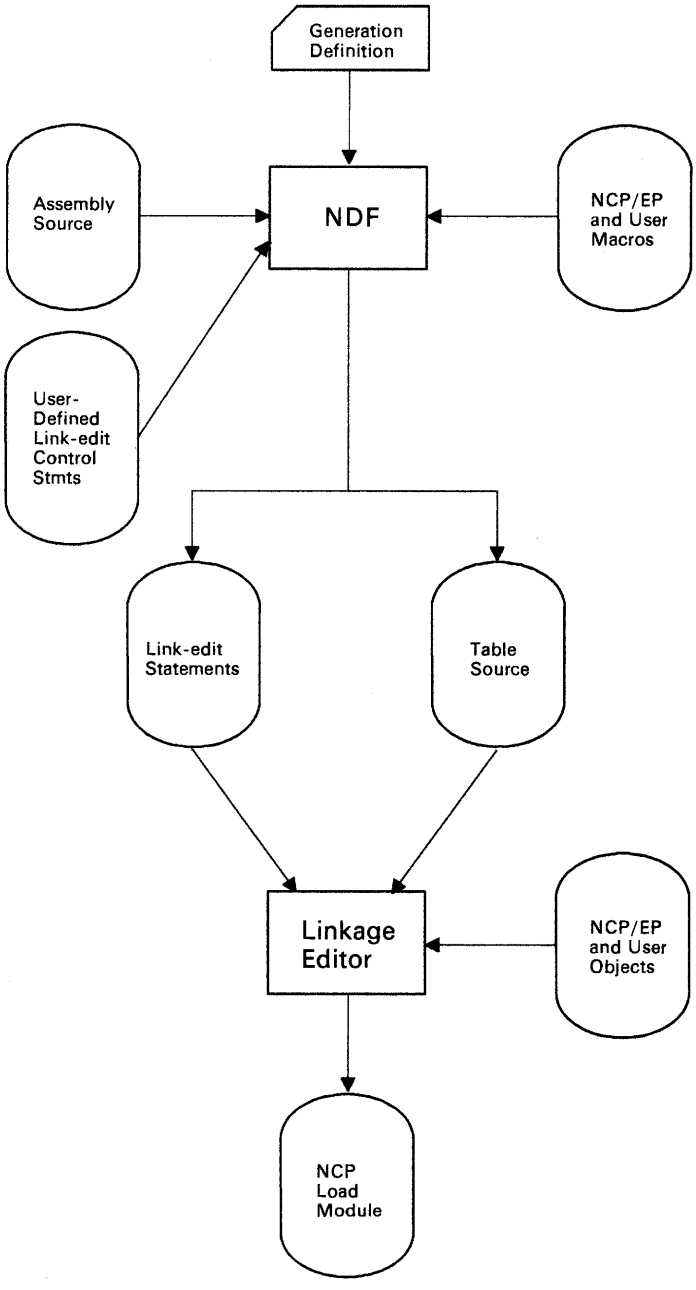


Figure 4-4. Generating User-Written Code and NCP without Using the NDF Standard Attachment Facility. This figure shows how user-written code is included in an NCP/PEP generation.

Before you generate, ensure that you:

- Assemble the user-written routines and code the link-edit control statements for the routines
- Code the appropriate operands on the GENEND definition statement for your user-written routines

- Place the members with SRCLO or SRCHI code in a macro library (SYSLIB) that is available to NDF
- Place all members that contain INCLUDE or ORDER link-edit control statements in a macro library in the NDF SYSLIB chain
- Place all macros in the NDF SYSLIB chain
- Modify the EXEC to include the SYSLIB chain, and the ULIB or user object code library ddname statement.

The generation validation phase of NDF reads the link-edit control statements and writes them into the same file as the standard NCP link-edit control statements.

For an example of an EXEC for generating user-written code and the NCP without using the standard attachment facility, see page 5-18.

Running a Dynamic Reconfiguration Generation

Use the text file from a dynamic reconfiguration generation to modify an NCP that is already running in a communication controller. Ensure that the original NCP was coded to allow dynamic reconfiguration. The dynamic reconfiguration generation produces a text file that is used by one of the access methods to modify the NCP.

A dynamic reconfiguration generation requires one table assembly and no link edit.

For an example of an EXEC for a dynamic reconfiguration generation, see page 5-19.

Understanding Listings and Error Messages

During generation validation, NDF creates a report that contains:

- The input statements interspersed with informational and error messages
- Keywords and statements passed to NDF from user-written generation load modules using the NDF standard attachment facility (**SSP V3R2 only**)
- A resource name and network address cross reference (only in valid runs)
- An error message summary
- A return code for the generation validation step.

Also, NDF creates a listing during the return code summary phase that gives return codes for the generation definition and for each of the table assemblies.

If any errors occur in generation validation, NDF notes these errors through diagnostic messages in the report. If these messages show serious errors (severity 4 or greater), NDF ends output of control-block source and link-edit control statements, but continues to validate the input definition statements. In this case, you must correct the errors and rerun. If no serious errors occur in NDF or the table assemblies, NDF runs to completion, the link edit is run, and a load module is produced.

Note: Other procedures, such as VTAM and CRP, require as input the same definition statements coded to generate the NCP, plus several additional operands and statements specific to each procedure. The additional operands and statements are identified in the NCP/SSP Resource Definition Reference. Although you can place these additional operands and statements in the NCP generation definition either before or after you generate the NCP, it is strongly recommended that you add them before, because executing the different procedures with different inputs can create errors.

If you are using the NDF standard attachment facility (SSP V3R2) to generate user-written code with your NCP, products that use the same generation definition may require as input the source statements and/or keywords passed to NDF from the user-written generation load modules. By specifying NEWDEFN= YES on the OPTIONS definition statement in your generation definition, and by specifying the NEWDEFN file in your EXEC, you instruct NDF to create a new generation definition for use by these procedures. This new generation definition contains both the NCP source statements and the statements passed from the user-written generation load modules.

NDF does not check the accuracy of the values coded for procedures, other than the NCP generation, that appear in the NCP generation

input. It checks only the keyword part for proper spelling. In the same way, these other procedures do not check the validity of the NCP definition statements and operands. Because of this, NDF requires that the NCP generation have no errors of severity 4 or greater.

Sample Generation Report

This section contains an example of an NDF generation report. Comments about the report are cited by numbers, for example **3** , and are not a part of the actual report. You can find the comments corresponding to these numbers following the report. Ellipses (. . .) in the report indicate that parts of the report were deleted for this example.

```
1                              2                    3
ACF SSP V3R2       01/20/86 18:30:03    DEFINITION SPECIFICATION                    PAGE 1
 4
LINE #                STATEMENT

    83  *         FULL LINE COMMENT 5
    84  PHY01     GROUP ECLTYPE=PHY          NTRI PHYSICAL GROUP 6
                    . . .
                      D               VIRTUAL=NO
                      D               TEXTTO=3.0  7
                      D               COMPACB=NO
    85  PHYLN1    LINE  ADDRESS=(80,FULL),UACB=WRONG,LOCADD=400001234567
                    . . .
                      D               MAXTSL=265
*WARNING  ICN021I  04  UACB=WRONG INVALID, INVALID NUMBER OF SUBOPERANDS, 2 EXPECTED WHEN EXLTYPE=PHYSICAL ON GROUP
8                     STATEMENT, REPLACED FOR STATEMENT KEYWORD VALIDATION
                    DELETED           UACB 9
                    ADDED             UACB(1)=X$P1AX 10
                    ADDED             UACB(2)=X$P1AR
                    G                 TYPE=NCP
                    G                 MAXPU=1
                    D                 CLOCKNG=EXT
                    D                 ATTACH=MODEM
 11                 . . .
GENERATED BY ECL
    86  X$P1SO    SERVICE
    87  ECLUPU1   PU
                    ADDED             ADDR(1)=01
                    G                 PUTYPE=1
                    D                 AVGPB=532
                    D                 ANS=STOP
                    D                 BNNSUP=()
                    D                 MAXDATA=78
                    . . .
    88  ECLLU1    LU
                    ADDED             LOCADDR(1)=0
                    D                 PACING(1)=1
                    D                 PACING(2)=1
                    D                 BATCH=NO
                    D                 LUDR=NO
                  . . .
ACF SSP V3         05/09/84 13:51:30    LABEL CROSS REFERENCE                       PAGE 6

                      LABEL CROSS REFERENCE -- SORTED BY LABEL NAME 12

LABEL         LINE      SA  ELEM       LABEL         LINE      SA  ELEM       LABEL         LINE      SA  ELEM

                  . . .

G14B1         178                      T14020BB      545       OE  001B       T14022G8      915       OE  004D
G14S1         362                      T14020BC      549       OE  001C       T14022G9      916       OE  004E
                  . . .
L14022        828       OE  0044       T14020B8      533       OE  0018       T14023A7      305       OE  0009
L14023        201       OE  0001       T14020B9      537       OE  0019       T14023A8      315       OE  000A
                  . . .
ACF SSP V3         05/09/84 13:51:30    LABEL CROSS REFERENCE                       PAGE 7

                      LABEL CROSS REFERENCE -- SORTED BY NETWORK ADDRESS 13

SA  ELEM    LINE    LABEL              SA  ELEM    LINE    LABEL              SA  ELEM    LINE    LABEL

OE  0001    201     L14023             OE  0038    747     T14020F8           OE  006E    90      DRPOOLLU
OE  0002    259     B14023A            OE  0039    749     T14020F9           OE  006F     6      NCPBUILD
OE  0003    269     T14023A1           OE  003A    767     T14020G            OE  0070     6      NCPBUILD
OE  0004    275     T14023A2           OE  003B    785     T14020G1           OE  0071     6      NCPBUILD
                  . . .
ACF SSP V3R2       11/20/85 18:30:03    ERROR SUMMARY 14                            PAGE 8

TOTAL MESSAGES     INFO     WARNING     ERROR     SEVERE     FATAL
      1              0          0         1          0          0

RETURN CODE IS 8

MESSAGES APPEAR AFTER THE LINES NUMBERED:

85

REGENERATION REQUIRED
```

Figure 4-5. Sample from an NDF Generation Report.

Comments

1 This portion of the header line on the report page contains the SSP version description.

2 This portion of the header line on the report page contains the date and time of the NDF run. This is the same date and time recorded in the date/time control block in the NCP/EP load module and printed in the formatted portion of the NCP/EP dump.

3 This portion of the header line on the report page identifies the report section. It has one of the following values: "DEFINITION SPECIFICATION," "LABEL CROSS REFERENCE," or "ERROR SUMMARY."

4 The line number column contains the line numbers of the generation definition listing.

5 NDF accepts full-line assembler comments.

6 This line contains a partial-line assembler comment.

7 This part of the report contains information describing operands that are defaulted or inherited.

A code prefixed to the associated message shows operands that are defaulted or operands using values from previous definition statements. These prefixes are:

- **G** Keyword inherited from GROUP
- **L** Keyword inherited from LINE
- **T** Keyword inherited from TERMINAL
- **C** Keyword inherited from CLUSTER
- **P** Keyword inherited from PU
- **D** Keyword that has been defaulted

8 Errors are indicated by an appropriate error number followed by a severity code and error message text. A severity code of 4 or more requires corrections to the generation definition before you can generate a load module. A severity code of less than 4 informs you that NDF has taken corrective action and does not require regeneration.

9 **(Only NCP V4R2 or NCP Subset)** A DELETE message indicates operands replaced by a user-written generation application or replaced by NDF for NTRI resources.

10 **(Only NCP V4R2 or NCP Subset)** An ADDED or APPENDED message indicates operands passed to NDF by a user-written generation application or added to the generation definition by NDF for NTRI resources. An operand is ADDED if the operand was not coded or if the operand was coded but then replaced. An operand is APPENDED if the operand was coded.

11 **(Only NCP V4R2 or NCP Subset)** "GENERATED BY ECL" or "GENERATED BY usergen name" precedes statements passed to NDF by user-written generation applications using the NDF standard attachment facility, or precedes statements added to the generation definition by NDF for NTRI resources.

12 The first label cross reference is sorted by label name. All user-coded labels appear in this list. If applicable, a network address is printed. Not all labels have associated network addresses. This section is not printed when severity codes of 4 or more exist. It is included in this sample for illustrative purposes only.

13 The second label cross reference is sorted by network address. Labels with no associated network addresses are omitted. This section is not printed when severity codes of 4 or more exist.

14 The error summary section contains an error count and a list of the line numbers immediately preceding error messages. If more than one error message immediately follows a given line, the line number is printed only once. If only informational messages follow a given line, an asterisk is printed next to the line number.

Chapter 5. Examples of EXECs for Generation under VM/SP

This chapter contains examples of EXECs written in the REXX language for generating your NCP under the VM/SP operating system. Most of these examples can also be found on the SSP tape sent from IBM Software Distribution and are supplied here for easy reference. However, the examples of EXECs for generating user-written code cannot be found on the SSP tape. Before using any examples, ensure that they match your operational environment.

This chapter includes examples of EXECs for the following types of generations:

- A FASTRUN generation
- An NCP/PEP generation with output written to disk
- An NCP/PEP generation with output written to tape
- An NCP/PEP generation with user-written code using the NDF standard attachment facility
- An NCP/PEP generation with user-written code without using the NDF standard attachment facility
- A dynamic reconfiguration generation.

Example for a FASTRUN Generation

To do a FASTRUN generation:

- Code FASTRUN=ON on the OPTIONS definition statement as the first executable statement in your generation definition, or code FASTRUN=ON as a parameter in your EXEC when calling NDF. This example uses the FASTRUN parameter coded in the EXEC.
- Ensure that your EXEC **does not** call the linkage editor. If the linkage editor step is present, an error condition results.
- **Do not** define the NCP macro library, since NDF does not run table assemblies for a FASTRUN generation. However, if you include user-written code in the generation definition, define the macro library that contains user-written link-edit control statements.

Following is an example of an EXEC for a FASTRUN generation:

```
/* EXAMPLE OF A REXX EXEC TO RUN A FASTRUN GENERATION                */
;
/* COPYRIGHT=NONE                                                    */
;
/* YOU MUST SUPPLY THE FILENAME AND FILETYPE OF THE INPUT GENERATION*/
/* DEFINITION ON THE COMMAND LINE WHEN INVOKING THE EXEC.            */
/*                                                                   */
/* YOU ARE ALLOWED TO SPECIFY THE FILEMODE OF YOUR INPUT GENERATION */
/* (WHICH DEFAULTS TO '*').  THIS PARAMETER IS OPTIONAL.             */
;
/* CORRECT FORM FOR INVOKING THE EXEC:                               */
/*     VMFAST FN=GEN_FN,FT=GEN_FT,FM=GEN_FM                          */
/*       -GEN_FN, GEN_FT, AND GEN_FM ARE VARIABLES WHICH YOU         */
/*        SUPPLY ACCORDING TO YOUR GENERATION                        */
/*       -ORDER OF PARAMETERS IS NOT IMPORTANT                       */
/*       -SPACES MAY BE USED INSTEAD OF COMMAS                       */
;
;
;
  ADDRESS COMMAND                    /* ENSURE CP/CMS ENVIRONMENT  */
  TRACE N
  GEN_FN=""
  GEN_FT=""                          /* INITIALIZE STRING VARIABLES*/
  GEN_FM=""
  ARG REST                           /* GET PARAMETERS FROM COMMAND*/
                                     /* LINE                       */
  REST=TRANSLATE(REST,' ',',')       /* GET RID OF COMMAS          */
  COUNT=WORDS(REST)
  LPCNT=1
```

This example is continued on the next page.

```
  DO WHILE LPCNT<=COUNT                  /* LOOP THROUGH ONCE FOR EACH */
    TEMP=WORD(REST,LPCNT)                /* WORD IN THE STRING         */
    PARSE VALUE TEMP WITH FRONT '=' BACK
    SELECT
      WHEN (ABBREV(TEMP,'FN')) THEN
        GEN_FN=BACK
      WHEN (ABBREV(TEMP,'FT')) THEN      /* SET APPROPRIATE VARIABLE   */
        GEN_FT=BACK                      /* ACCORDING TO THE ASSIGNMENT*/
      WHEN (ABBREV(TEMP,'FM')) THEN      /* MADE ON THE COMMAND LINE   */
        GEN_FM=BACK
      OTHERWISE
        SAY TEMP" IS NOT VALID, IGNORED"
    END                                  /* END SELECT                 */
    LPCNT=LPCNT+1
  END                                    /* END DO                     */
  IF GEN_FM="" THEN                      /* DEFAULT FILETYPE TO "*" IF */
    GEN_FM="*"                           /* NOT CODED                  */
;
;
/* SEE IF GEN_FN AND GEN_FT WERE PASSED ON COMMAND LINE.                */
/* IF GEN NAME WAS NOT SPECIFIED, GIVE CORRECT FORM AND EXIT.           */
;
  IF (GEN_FN="")|(GEN_FT="") THEN
    DO
      SAY "CORRECT FORM:"
      SAY ""
      SAY "   VMFAST FN=GEN_FN,FT=GEN_FT,FM=GEN_FM"
      SAY ""
      SAY "     -GEN_FN, GEN_FT, AND GEN_FM ARE VARIABLES WHICH"
      SAY "      YOU SUPPLY ACCORDING TO YOUR GENERATION"
      SAY "     -ORDER OF PARAMETERS IS NOT IMPORTANT"
      SAY "     -SPACES MAY BE USED INSTEAD OF COMMAS"
      SAY "     -IF OMITTED, THE DEFAULT FOR GEN_FM IS '*'"
      SAY "     -GEN_FN AND GEN_FT ARE REQUIRED"
      EXIT
    END
;
  'STATE' GEN_FN GEN_FT GEN_FM         /* SEE IF GEN EXISTS ON DISK  */
  IF RC ¬ = 0 THEN
    DO
      SAY GEN_FN GEN_FT GEN_FM "DOES NOT EXIST"
      EXIT RC                            /* EXIT IF GEN DOESN'T EXIST  */
    END
;
;
/* CLEAR OLD FILE DEFINITIONS                                          */
  'FILEDEF * CLEAR'
/* WORKING SPILL FILE                                                  */
/* THE DBWORKFL IS NEEDED ONLY WHEN THERE IS NOT ENOUGH VIRTUAL        */
/* MEMORY TO HOLD ALL OF NDF'S WORK DATA OR IF USERGEN IS SPECIFIED.*/
/*'FILEDEF DBWORKFL DISK  DBWORKFL FILE A ( XTENT 40'                  */
/* INPUT FILE WITH NCP/EP GENERATION STATEMENTS                        */
  'FILEDEF GENDECK DISK' GEN_FN GEN_FT GEN_FM
/* GENERATION VALIDATION STEP OUTPUT                                   */
  'FILEDEF SYSPRINT DISK' GEN_FN 'LISTING A'
/* NDF SUMMARY LISTING                                                 */
  'FILEDEF PRINTER TERM'
/* RUN THE NDF STEP                                                    */
  'ICNRTNDF (FASTRUN(ON))'
  EXIT RC
```

Example for an NCP/PEP Generation with Output Written to Disk

This example shows an EXEC used to generate an NCP load module for the IBM 3725 Communication Controller with output files written to disk. For an IBM 3705 or 3720 Communication Controller:

- The EXEC is slightly different.
- The NCP macro library used for the NDF job step may be different.
- The ddname for the library of preassembled NCP object modules in the link-edit step may be different.

These differences are explained in the text included in the following example.

Note: When running the link-edit step for a standard NCP/PEP generation specify the ALIGN2 option in the EXEC in the link-edit step to ensure that certain control sections within the load module are aligned on 2K boundaries. If this is not specified, the default is alignment on 4K page boundaries, which may use excessive controller storage.

Note for NTRI Users: You must code NEWDEFN=YES on the OPTIONS definition statement as the first executable statement in your generation definition and define the NEWDEFN file in your EXEC.

Following is an example for an NCP/PEP generation with output written to disk:

```
/* EXAMPLE OF AN EXEC TO RUN A NCP/PEP GENERATION WITH ALL OUTPUT   */
/* FILES WRITTEN TO DISK                                            */
;
/* COPYRIGHT=NONE                                                   */
;
/* THIS SAMPLE CAN BE USED TO GENERATE AN NCP WITH IBM SPECIAL      */
/* PRODUCTS IF THE MACROS AND OBJECT MODULES FOR THESE PRODUCTS HAVE*/
/* BEEN MERGED INTO THE APPROPRIATE NCP LIBRARIES.                  */
;
;
/* YOU MUST SUPPLY THE FILENAME AND FILETYPE OF THE INPUT GENERATION*/
/* DECK ON THE COMMAND LINE WHEN INVOKING THE EXEC.  YOU MUST ALSO  */
/* SUPPLY THE MODEL NUMBER OF THE CONTROLLER.                       */
/*                                                                  */
/* YOU ARE ALLOWED TO SPECIFY THE FILEMODE OF YOUR INPUT GENERATION */
/* (WHICH DEFAULTS TO '*'), VERSION OF THE GENERATION (WHICH        */
/* DEFAULTS TO 'V3'), AND TEST (WHICH DEFAULTS TO 'T=NO').          */
/* THESE PARAMETERS ARE OPTIONAL.                                   */
;
/* CORRECT FORM FOR INVOKING THE EXEC:                              */
/*     VMNCP FN=GEN_FN,FT=GEN_FT,FM=GEN_FM,V=VERSION,M=MODEL,T=NO   */
/*       -GEN_FN, GEN_FT, GEN_FM, VERSION, AND MODEL ARE VARIABLES  */
/*        WHICH YOU SUPPLY ACCORDING TO YOUR GENERATION             */
/*       -YOU MAY CODE 'T=YES' FOR A RUN ON TEST NCP LIBRARIES      */
```

This example is continued on the next page.

```
/*       -ORDER OF PARAMETERS IS NOT IMPORTANT                          */
/*       -SPACES MAY BE USED INSTEAD OF COMMAS                          */
/*       -FOR NCP SUBSET, CODE V=V4S                                    */
;
;
/* THE ASSIGNMENT OF THE MACRO AND OBJECT LIBRARY NAMES IS DERIVED     */
/* FROM THE PARAMETERS PASSED ON THE COMMAND LINE; THEY MAY RESIDE     */
/* ON ANY ACCESSED DISK.                                               */
/*                                                                     */
/* VARIABLES ARE DEFINED AS FOLLOWS:                                   */
/*      MACRO = MACLIB NAME                                            */
/*      OBJECT = OBJLIB NAME                                           */
;
/*********************************************************************/
/* THE FOLLOWING TABLE WILL AID YOU IN WHICH MACRO AND OBJECT          */
/* LIBRARIES CORRESPOND TO A PARTICULAR MODEL AND VERSION.  IF YOUR    */
/* LIBRARY NAMES DO NOT AGREE WITH THIS TABLE, YOU WILL NEED TO        */
/* SUBSTITUTE YOUR LIBRARY NAME FOR THE STANDARD LIBRARY NAME WHERE    */
/* APPROPRIATE.  YOU CAN USE THE "ALL" COMMAND ON THE FOLLOWING        */
/* STRINGS TO FIND LINES TO CHANGE FOR A PARTICULAR VERSION & MODEL.*/
/*                                                                     */
/* 'T=YES' IS ALLOWED FOR TESTING TO FACILITATE LIBRARY MAINTENANCE.*/
/* (YOU MUST GO TO THE APPROPRIATE PLACE AND KEY IN YOUR TEST-LIB      */
/*  NAMES TO USE T=YES.)                                               */
/* T=YES WILL RUN WITH ANY MODEL AND VERSION (YOU MUST CODE MODEL).    */
/*                                                                     */
/*     ALLTAG1 - TEST                                                  */
/*     ALLTAG2 - V3 & 3705                                             */
/*     ALLTAG3 - V3 & 3725                                             */
/*     ALLTAG4 - V4R2 & 3725 OR 3720                                   */
/*     ALLTAG5 - V4 SUBSET & 3720                                      */
/*                                                                     */
/*                          M O D E L                                  */
/*                                                                     */
/*                 3705          3725          3720                  */
/*               ---------------------------------------               */
/*     V3      |  MAC3705    |  MAC3725    |  NOT        |             */
/*  V          |  OBJ3705    |  OBJ3725    |  SUPPORTED  |             */
/*  E          |-------------------------------------------|           */
/*  R  V4      |  NOT        |  NOT        |  NOT        |             */
/*  S          |  SUPPORTED  |  SUPPORTED  |  SUPPORTED  |             */
/*  I          |-------------------------------------------|           */
/*  O  V4R2    |  NOT        |  MAC3725    |  MAC3725    |             */
/*  N          |  SUPPORTED  |  OBJ3725    |  OBJ3725    |             */
/*             |-------------------------------------------|           */
/*     V4      |  NOT        |  NOT        |  MAC3725    |             */
/*     SUBSET  |  SUPPORTED  |  SUPPORTED  |  OBJ3725    |             */
/*               ---------------------------------------               */
/*********************************************************************/
;
  ADDRESS COMMAND                    /* ENSURE CP/CMS ENVIRONMENT  */
  TRACE N
  GEN_FN=""
  GEN_FT=""                          /* INITIALIZE STRING VARIABLES*/
  GEN_FM=""
  VERSION=""
  MODEL=""
  T="NO"
  ARG REST                           /* GET PARAMETERS FROM COMMAND*/
                                     /* LINE                       */
```

This example is continued on the next page.

```
  REST=TRANSLATE(REST,' ',',')          /* GET RID OF COMMAS          */
  COUNT=WORDS(REST)
  LPCNT=1
  DO WHILE LPCNT<=COUNT                 /* LOOP THROUGH ONCE FOR EACH */
    TEMP=WORD(REST,LPCNT)               /* WORD IN THE STRING         */
    PARSE VALUE TEMP WITH FRONT '=' BACK
    SELECT
      WHEN (ABBREV(TEMP,'FN')) THEN
        GEN_FN=BACK
      WHEN (ABBREV(TEMP,'FT')) THEN     /* SET APPROPRIATE VARIABLE   */
        GEN_FT=BACK                     /* ACCORDING TO THE ASSIGNMENT*/
      WHEN (ABBREV(TEMP,'FM')) THEN     /* MADE ON THE COMMAND LINE   */
        GEN_FM=BACK
      WHEN (ABBREV(TEMP,'V')) THEN
        VERSION=BACK
      WHEN (ABBREV(TEMP,'M')) THEN
        MODEL=BACK
      WHEN (ABBREV(TEMP,'T')) THEN
        T=BACK
      OTHERWISE
        SAY TEMP" IS NOT VALID, IGNORED"
    END                                 /* END SELECT                 */
    LPCNT=LPCNT+1
  END                                   /* END DO                     */
  IF GEN_FM="" THEN                     /* DEFAULT FILETYPE TO "*" IF */
    GEN_FM="*"                          /* NOT CODED                  */
;
;
/* SEE IF GEN_FN, GEN_FT, AND MODEL WERE PASSED ON COMMAND LINE.     */
/* IF GEN NAME OR MODEL WERE NOT SPECIFIED GIVE CORRECT FORM & EXIT.*/
;
  IF (GEN_FN="")|(GEN_FT="")|(MODEL="") THEN
    DO
      SAY "CORRECT FORM:"
      SAY ""
      SAY "VMNCP FN=GEN_FN,FT=GEN_FT,FM=GEN_FM,V=VERSION,M=MODEL,T=NO"
      SAY ""
      SAY "      -GEN_FN, GEN_FT, GEN_FM, VERSION, MODEL ARE VARIABLES"
      SAY "       WHICH YOU SUPPLY ACCORDING TO YOUR GENERATION"
      SAY "      -YOU MAY CODE 'T=YES' FOR A RUN ON TEST NCP LIBRARIES"
      SAY "      -ORDER OF PARAMETERS IS NOT IMPORTANT"
      SAY "      -SPACES MAY BE USED INSTEAD OF COMMAS"
      SAY "      -FOR NCP SUBSET, CODE V=V4S"
      SAY "      -IF OMITTED, DEFAULTS ARE:"
      SAY "         FM=*"
      SAY "         V=V3"
      SAY "         T=NO"
      SAY "      -GEN_FN, GEN_FT, AND MODEL ARE REQUIRED"
      EXIT
    END
;
;
  'STATE' GEN_FN GEN_FT GEN_FM          /* SEE IF GEN EXISTS ON DISK  */
  IF RC ¬ = 0 THEN
    DO
      SAY GEN_FN GEN_FT GEN_FM "DOES NOT EXIST"
      EXIT RC                           /* EXIT IF GEN DOESN'T EXIST  */
    END
```

This example is continued on the next page.

```
;
;
/* THIS STRUCTURE VALIDATES VERSION AND MODEL AS TAKEN FROM THE     */
/* COMMAND LINE AND SETS "MACRO" AND "OBJECT" ACCORDINGLY.          */
/*                                                                  */
/* NOTE:  IF YOU HAVE CHANGED A LIBRARY NAME, OR YOUR LIBRARY NAMES */
/* DO NOT AGREE WITH THE TABLE ABOVE, CHANGE THE ASSIGNMENT         */
/* STATEMENTS OF "MACRO" AND "OBJECT" TO REFLECT YOUR LIBRARY NAMES */
/* IN THE APPROPRIATE PLACE IN THIS STRUCTURE (ACCORDING TO YOUR    */
/* VERSION AND MODEL).                                              */
;
;
  IF (VERSION="") THEN                  /* VERSION DEFAULTS TO V3     */
    DO                                  /* IF NOT CODED               */
      SAY "DEFAULTING TO VERSION = V3"
      VERSION="V3"
    END
  SELECT
    WHEN (T="YES") THEN
      DO
        MACRO=MACXXXX                   /* FOR TEST            ALLTAG1*/
        OBJECT=OBJXXXX                  /*                     ALLTAG1*/
      END
    WHEN (VERSION="V3")&(MODEL="3705") THEN
      DO
        MACRO=MAC3705                   /* FOR V3 & 3705       ALLTAG2*/
        OBJECT=OBJ3705                  /*                     ALLTAG2*/
      END
    WHEN (VERSION="V3")&(MODEL="3725") THEN
      DO
        MACRO=MAC3725                   /* FOR V3 & 3725       ALLTAG3*/
        OBJECT=OBJ3725                  /*                     ALLTAG3*/
      END
    WHEN (VERSION="V4R2")&((MODEL="3725")|(MODEL="3720")) THEN
      DO
        MACRO=MAC3725                   /* FOR V4R2&3725|3720  ALLTAG4*/
        OBJECT=OBJ3725                  /*                     ALLTAG4*/
      END
    WHEN (VERSION="V4S")&(MODEL="3720") THEN
      DO
        MACRO=MAC3725                   /* FOR V4 SUBSET       ALLTAG5*/
        OBJECT=OBJ3725                  /* & 3720              ALLTAG5*/
      END
    OTHERWISE
      DO
        SAY "VERSION = "VERSION" NOT VALID WITH MODEL = "MODEL
        SAY "WHEN RUNNING UNDER VM"
        EXIT
      END
  END                                   /* END SELECT                 */
;
;
  'ESTATE' MACRO 'MACLIB *'             /* SEE IF MACLIB EXISTS       */
  IF RC ¬ = 0 THEN
    SAY "ERROR IN ACCESSING" MACRO "MACLIB"
```

This example is continued on the next page.

```
;
;
/* CLEAR OLD FILE DEFINITIONS                                          */
  'FILEDEF * CLEAR'
/* WORKING SPILL FILE                                                  */
/* THE DBWORKFL IS NEEDED ONLY WHEN THERE IS NOT ENOUGH VIRTUAL        */
/* MEMORY TO HOLD ALL OF NDF'S WORK DATA OR IF USERGEN IS SPECIFIED.*/
/*'FILEDEF DBWORKFL DISK DBWORKFL FILE A ( XTENT 40'                  */
/* MACRO LIBRARIES USED IN THE TABLE ASSEMBLY PHASE OF NDF             */
  'FILEDEF SYSLIB DISK' MACRO 'MACLIB *'
  'GLOBAL MACLIB' MACRO
/* INPUT FILE WITH NCP/EP GENERATION STATEMENTS                        */
  'FILEDEF GENDECK DISK' GEN_FN GEN_FT GEN_FM
/* GENERATION VALIDATION STEP OUTPUT                                   */
  'FILEDEF SYSPRINT DISK' GEN_FN 'LISTING A'
/* NDF SUMMARY LISTING                                                 */
  'FILEDEF PRINTER TERM'
/* SOURCE FOR TABLE 1 ASSEMBLY - OUTPUT FROM GENERATION VALIDATION     */
  'FILEDEF TBL1SRCE DISK TABLE1 SOURCE A'
/* LISTING FROM THE TABLE1 ASSEMBLY                                    */
  'FILEDEF TBL1LIST DISK TABLE1 LISTING A'
/* TEXT OUTPUT FROM THE TABLE1 ASSEMBLY                                */
  'FILEDEF TBL1OBJ DISK TABLE1 TEXT A'
/* SOURCE FOR TABLE 2 ASSEMBLY - OUTPUT FROM GENERATION VALIDATION     */
  'FILEDEF TBL2SRCE DISK TABLE2 SOURCE A'
/* LISTING FROM THE TABLE2 ASSEMBLY                                    */
  'FILEDEF TBL2LIST DISK TABLE2 LISTING A'
/* TEXT OUTPUT FROM THE TABLE2 ASSEMBLY                                */
  'FILEDEF TBL2OBJ DISK TABLE2 TEXT A'
/* LINK EDIT CARDS OUTPUT FROM THE GENERATION VALIDATION STEP          */
  'FILEDEF LNKSTMT DISK NCPINCL TEXT A'
/* TEMPORARY WORK FILE USED BY THE TABLE ASSEMBLIES                    */
  'FILEDEF SYSUT1 DISK SYSUT1 TEMP A4 (BLOCK 4000'
/* RUN THE NDF STEP                                                    */
  'ICNRTNDF'
/* EXIT BECAUSE OF AN ERROR DURING GENERATION VALIDATION               */
  IF RC ¬ = 0 THEN
    DO
      SAY "***ERROR IN EXECUTING NDF***"
      EXIT RC
    END
/* PUT TEXT OUTPUT FROM TABLE ASSEMBLIES INTO A SIMULATED PDS          */
  'TXTLIB GEN OBJ TABLE1 TABLE2'
  IF RC ¬ = 0 THEN
    DO
      SAY "CANNOT FIND TABLE1 TEXT OR TABLE2 TEXT"
      EXIT 99
    END
/* ERASE TEMPORARY FILES                                               */
  'ERASE SYSUT1 TEMP A'
  'ERASE TABLE1 SOURCE'
  'ERASE TABLE2 SOURCE'
  'ERASE TABLE1 TEXT'
  'ERASE TABLE2 TEXT'
;
```

This example is continued on the next page.

```
  'ESTATE' OBJECT 'TXTLIB *'          /* SEE IF OBJLIB EXISTS       */
  IF RC ¬ = 0 THEN
    SAY "ERROR IN ACCESSING" OBJECT "TXTLIB"
;
/* FILDEFS FOR THE LINK EDIT STEP                                    */
  'FILEDEF SYSUT1 CLEAR'
  'FILEDEF' OBJECT 'DISK' OBJECT 'TXTLIB *'
/* NCP/EP TABLE TEXT                                                 */
  'FILEDEF SYSPUNCH DISK OBJ TXTLIB A'
/* NAME OF OUTPUT LIBRARY FOR THE LOAD MODULE                        */
  'FILEDEF SYSLMOD DISK' GEN_FN 'LOADLIB A'
/* RUN LINKAGE EDITOR                                                */
  'LKED NCPINCL (MAP NCAL NOTERM LET LIST ALIGN2'
  EXIT RC
```

Example for an NCP/PEP Generation with Output Written to Tape

If you detect an error while you are generating or running your NCP, you might want to write certain listing files to tape. This example shows an EXEC for generating an NCP load module for an IBM 3725 Communication Controller with listing files from the Table 1 assembly, the Table 2 assembly, and the link edit written to tape. For an IBM 3705 or 3720 Communication Controller:

- The EXEC is slightly different.
- The NCP macro library used for the NDF job step may be different.
- The ddname for the library of preassembled NCP object modules in the link-edit step may be different.

These differences are explained in the text included in the following example.

Note: When running the link-edit step for a standard NCP/PEP generation specify the ALIGN2 option in the EXEC in the link-edit step to ensure that certain control sections within the load module are aligned on 2K boundaries. If this is not specified, the default is alignment on 4K page boundaries, which may use excessive controller storage.

Note for NTRI Users: You must code NEWDEFN=YES on the OPTIONS definition statement as the first executable statement in your generation definition and define the NEWDEFN file in your EXEC.

Following is an example for an NCP/PEP generation with output written to tape:

```
/* EXAMPLE OF AN EXEC TO RUN A NCP/PEP GENERATION WITH SOME OUTPUT  */
/* FILES WRITTEN TO TAPE                                            */
;
/* COPYRIGHT=NONE                                                   */
;
/* THIS SAMPLE CAN BE USED TO GENERATE AN NCP WITH IBM SPECIAL      */
/* PRODUCTS IF THE MACROS AND OBJECT MODULES FOR THOSE PRODUCTS     */
/* HAVE BEEN MERGED INTO THE APPROPRIATE NCP LIBRARIES.             */
;
/* THE TABLE 1 LISTING AND THE TABLE 2 LISTING ARE WRITTEN TO TAPE  */
/* AND ONLY COPIED TO DISK WHEN AN ERROR IS DETECTED DURING THE     */
/* TABLE ASSEMBLIES.                                                */
;
;
/* YOU MUST SUPPLY THE FILENAME AND FILETYPE OF THE INPUT GENERATION*/
/* DEFINITION ON THE COMMAND LINE WHEN INVOKING THE EXEC.           */
/* YOU MUST ALSO SUPPLY THE MODEL NUMBER OF THE CONTROLLER.         */
/*                                                                  */
/* YOU ARE ALLOWED TO SPECIFY THE FILEMODE OF YOUR INPUT GENERATION */
/* (WHICH DEFAULTS TO '*'), VERSION OF YOUR GENERATION (WHICH       */
/* (WHICH DEFAULTS TO 'V3'), AND TEST (WHICH DEFAULTS TO 'T=NO').   */
/* THESE PARAMETERS ARE OPTIONAL.                                   */
```

This example is continued on the next page.

```
;
/* CORRECT FORM FOR INVOKING THE EXEC:                                  */
/*     VMTAPE FN=GEN_FN,FT=GEN_FT,FM=GEN_FM,V=VERSION,M=MODEL,T=NO      */
/*       -GEN_FN, GEN_FT, GEN_FM, VERSION, AND MODEL ARE VARIABLES      */
/*        WHICH YOU SUPPLY ACCORDING TO YOUR GENERATION                 */
/*       -YOU MAY CODE 'T=YES' FOR A RUN ON TEST NCP LIBRARIES          */
/*       -ORDER OF PARAMETERS IS NOT IMPORTANT                          */
/*       -SPACES MAY BE USED INSTEAD OF COMMAS                          */
/*       -FOR NCP SUBSET, CODE V=V4S                                    */
;
;
/* THE ASSIGNMENT OF THE MACRO AND OBJECT LIBRARY NAMES IS DERIVED      */
/* FROM THE PARAMETERS PASSED ON THE COMMAND LINE; THEY MAY RESIDE      */
/* ON ANY ACCESSED DISK.                                                */
/*                                                                      */
/* VARIABLES ARE DEFINED AS FOLLOWS:                                    */
/*      MACRO = MACLIB NAME                                             */
/*      OBJECT = OBJLIB NAME                                            */
;
/**********************************************************************/
/* THE FOLLOWING TABLE WILL AID YOU IN WHICH MACRO AND OBJECT           */
/* LIBRARIES CORRESPOND TO A PARTICULAR MODEL AND VERSION.  IF YOUR     */
/* LIBRARY NAMES DO NOT AGREE WITH THIS TABLE, YOU WILL NEED TO         */
/* SUBSTITUTE YOUR LIBRARY NAME FOR THE STANDARD LIBRARY NAME WHERE     */
/* APPROPRIATE.  YOU CAN USE THE "ALL" COMMAND ON THE FOLLOWING         */
/* STRINGS TO FIND LINES TO CHANGE FOR A PARTICULAR VERSION & MODEL.    */
/*                                                                      */
/* 'T=YES' IS ALLOWED FOR TESTING TO FACILITATE LIBRARY MAINTENANCE.    */
/* (YOU MUST GO TO THE APPROPRIATE PLACE AND KEY IN YOUR TEST-LIB       */
/*  NAMES TO USE T=YES.)                                                */
/* T=YES WILL RUN WITH ANY MODEL AND VERSION (YOU MUST CODE MODEL).     */
/*                                                                      */
/*     ALLTAG1 - TEST                                                   */
/*     ALLTAG2 - V3 & 3705                                              */
/*     ALLTAG3 - V3 & 3725                                              */
/*     ALLTAG4 - V4R2 & 3725 OR 3720                                    */
/*     ALLTAG5 - V4 SUBSET & 3720                                       */
/*                                                                      */
/*                           M O D E L                                  */
/*                                                                      */
/*                  3705          3725          3720                  */
/*                 ------------------------------------                 */
/*     V3         |  MAC3705    |  MAC3725    |  NOT        |           */
/*  V              |  OBJ3705    |  OBJ3725    |  SUPPORTED  |           */
/*  E              |-------------------------------------|              */
/*  R  V4          |  NOT        |  NOT        |  NOT        |           */
/*  S              |  SUPPORTED  |  SUPPORTED  |  SUPPORTED  |           */
/*  I              |-------------------------------------|              */
/*  O  V4R2        |  NOT        |  MAC3725    |  MAC3725    |           */
/*  N              |  SUPPORTED  |  OBJ3725    |  OBJ3725    |           */
/*                 |-------------------------------------|              */
/*     V4         |  NOT        |  NOT        |  MAC3725    |           */
/*     SUBSET     |  SUPPORTED  |  SUPPORTED  |  OBJ3725    |           */
/*                 ------------------------------------                 */
/**********************************************************************/
```

This example is continued on the next page.

```
;
  ADDRESS COMMAND                          /* ENSURE CP/CMS ENVIRONMENT  */
  TRACE N
  GEN_FN=""
  GEN_FT=""                                /* INITIALIZE STRING VARIABLES*/
  GEN_FM=""
  VERSION=""
  MODEL=""
  T="NO"
  ARG REST                                 /* GET PARAMETERS FROM COMMAND*/
                                           /* LINE                       */
  REST=TRANSLATE(REST,' ',',')             /* GET RID OF COMMAS          */
  COUNT=WORDS(REST)
  LPCNT=1
  DO WHILE LPCNT<=COUNT                    /* LOOP THROUGH ONCE FOR EACH */
    TEMP=WORD(REST,LPCNT)                  /* WORD IN THE STRING         */
    PARSE VALUE TEMP WITH FRONT '=' BACK
    SELECT
      WHEN (ABBREV(TEMP,'FN')) THEN
        GEN_FN=BACK
      WHEN (ABBREV(TEMP,'FT')) THEN        /* SET APPROPRIATE VARIABLE   */
        GEN_FT=BACK                        /* ACCORDING TO THE ASSIGNMENT*/
      WHEN (ABBREV(TEMP,'FM')) THEN        /* MADE ON THE COMMAND LINE   */
        GEN_FM=BACK
      WHEN (ABBREV(TEMP,'V')) THEN
        VERSION=BACK
      WHEN (ABBREV(TEMP,'M')) THEN
        MODEL=BACK
      WHEN (ABBREV(TEMP,'T')) THEN
        T=BACK
      OTHERWISE
        SAY TEMP" IS NOT VALID, IGNORED"
    END                                    /* END SELECT                 */
    LPCNT=LPCNT+1
  END                                      /* END DO                     */
  IF GEN_FM="" THEN                        /* DEFAULT FILETYPE TO "*" IF */
    GEN_FM="*"                             /* NOT CODED                  */
;
;
/* SEE IF GEN_FN, GEN_FT, AND MODEL WERE PASSED ON COMMAND LINE.    */
/* IF GEN NAME OR MODEL WERE NOT SPECIFIED GIVE CORRECT FORM & EXIT.*/
```

This example is continued on the next page.

```
;
  IF (GEN_FN="")|(GEN_FT="")|(MODEL="") THEN
    DO
     SAY "CORRECT FORM:"
     SAY ""
     SAY "VMTAPE FN=GEN_FN,FT=GEN_FT,FM=GEN_FM,V=VERSION,M=MODEL,T=NO"
     SAY ""
     SAY "      -GEN_FN, GEN_FT, GEN_FM, VERSION, MODEL ARE VARIABLES"
     SAY "       WHICH YOU SUPPLY ACCORDING TO YOUR GENERATION"
     SAY "      -YOU MAY CODE 'T=YES' FOR A RUN ON TEST NCP LIBRARIES"
     SAY "      -ORDER OF PARAMETERS IS NOT IMPORTANT"
     SAY "      -SPACES MAY BE USED INSTEAD OF COMMAS"
     SAY "      -FOR NCP SUBSET, CODE V=V4S"
     SAY "      -IF OMITTED, DEFAULTS ARE:"
     SAY "         FM=*"
     SAY "         V=V3"
     SAY "         T=NO"
     SAY "      -GEN_FN, GEN_FT, AND MODEL ARE REQUIRED"
     EXIT
    END
;
;
  'STATE' GEN_FN GEN_FT GEN_FM        /* SEE IF GEN EXISTS ON DISK  */
  IF RC ¬ = 0 THEN
    DO
      SAY GEN_FN GEN_FT GEN_FM "DOES NOT EXIST"
      EXIT RC                         /* EXIT IF GEN DOESN'T EXIST  */
    END
;
;
/* THIS STRUCTURE VALIDATES VERSION AND MODEL AS TAKEN FROM THE     */
/* COMMAND LINE AND SETS "MACRO" AND "OBJECT" ACCORDINGLY.          */
/*                                                                  */
/* NOTE:  IF YOU HAVE CHANGED A LIBRARY NAME, OR YOUR LIBRARY NAMES */
/* DO NOT AGREE WITH THE TABLE ABOVE, CHANGE THE ASSIGNMENT         */
/* STATEMENTS OF "MACRO" AND "OBJECT" TO REFLECT YOUR LIBRARY NAMES */
/* IN THE APPROPRIATE PLACE IN THIS STRUCTURE (ACCORDING TO YOUR    */
/* VERSION AND MODEL).                                              */
```

This example is continued on the next page.

```
;
;
  IF (VERSION="") THEN                    /* VERSION DEFAULTS TO V3      */
    DO                                    /* IF NOT CODED                */
      SAY "DEFAULTING TO VERSION = V3"
      VERSION="V3"
    END
  SELECT
    WHEN (T="YES") THEN
      DO
        MACRO=MACXXXX                     /* FOR TEST            ALLTAG1*/
        OBJECT=OBJXXXX                    /*                     ALLTAG1*/
      END
    WHEN (VERSION="V3")&(MODEL="3705") THEN
      DO
        MACRO=MAC3705                     /* FOR V3 & 3705       ALLTAG2*/
        OBJECT=OBJ3705                    /*                     ALLTAG2*/
      END
    WHEN (VERSION="V3")&(MODEL="3725") THEN
      DO
        MACRO=MAC3725                     /* FOR V3 & 3725       ALLTAG3*/
        OBJECT=OBJ3725                    /*                     ALLTAG3*/
      END
    WHEN (VERSION="V4R2")&((MODEL="3725")|(MODEL="3720")) THEN
      DO
        MACRO=MAC3725                     /* FOR V4R2&3725|3720  ALLTAG4*/
        OBJECT=OBJ3725                    /*                     ALLTAG4*/
      END
    WHEN (VERSION="V4S")&(MODEL="3720") THEN
      DO
        MACRO=MAC3725                     /* FOR V4 SUBSET       ALLTAG5*/
        OBJECT=OBJ3725                    /* & 3720              ALLTAG5*/
      END
    OTHERWISE
      DO
        SAY "VERSION = "VERSION" NOT VALID WITH MODEL = "MODEL
        SAY "WHEN RUNNING UNDER VM"
        EXIT
      END
  END                                     /* END SELECT                  */
;
;
  'ESTATE' MACRO 'MACLIB *'               /* SEE IF MACLIB EXISTS        */
  IF RC ¬ = 0 THEN
    SAY "ERROR IN ACCESSING" MACRO "MACLIB"
;
;
  'TAPE REW'
   IF RC ¬ = 0 THEN
     DO
       SAY "ERROR IN ACCESSING TAPE"
       EXIT RC
     END
  'TAPE WVOL1 TAPE1'
/* CLEAR OLD FILE DEFINITIONS                                            */
  'FILEDEF * CLEAR'
```

This example is continued on the next page.

```
/* WORKING SPILL FILE                                                  */
/* THE DBWORKFL IS NEEDED ONLY WHEN THERE IS NOT ENOUGH VIRTUAL        */
/* MEMORY TO HOLD ALL OF NDF'S WORK DATA OR IF USERGEN IS SPECIFIED.*/
/*'FILEDEF DBWORKFL DISK  DBWORKFL FILE A ( XTENT 40'                 */
/* MACRO LIBRARIES USED IN THE TABLE ASSEMBLY PHASE OF NDF            */
  'FILEDEF SYSLIB DISK' MACRO 'MACLIB *'
  'GLOBAL MACLIB' MACRO
/* INPUT FILE WITH NCP/EP GENERATION STATEMENTS                       */
  'FILEDEF GENDECK DISK' GEN_FN GEN_FT GEN_FM
/* GENERATION VALIDATION STEP OUTPUT                                  */
  'FILEDEF SYSPRINT DISK' GEN_FN 'LISTING A'
/* NDF SUMMARY LISTING                                                */
  'FILEDEF PRINTER TERM'
/* SOURCE FOR TABLE 1 ASSEMBLY - OUTPUT FROM GENERATION VALIDATION    */
  'FILEDEF TBL1SRCE DISK TABLE1 SOURCE A'
/* LISTING FROM THE TABLE1 ASSEMBLY                                   */
  'FILEDEF TBL1LIST TAP1 SL  ( BLKSIZE 7260 LRECL 121 RECFM FB'
/* TEXT OUTPUT FROM THE TABLE1 ASSEMBLY                               */
  'FILEDEF TBL1OBJ DISK TABLE1 TEXT A'
/* SOURCE FOR TABLE 2 ASSEMBLY - OUTPUT FROM GENERATION VALIDATION    */
  'FILEDEF TBL2SRCE DISK TABLE2 SOURCE A'
/* LISTING FROM THE TABLE2 ASSEMBLY                                   */
  'FILEDEF TBL2LIST TAP1 SL (LEAVE BLKSIZE 7260 LRECL 121 RECFM FB'
/* TEXT OUTPUT FROM THE TABLE2 ASSEMBLY                               */
  'FILEDEF TBL2OBJ DISK TABLE2 TEXT A'
/* LINK EDIT CARDS OUTPUT FROM THE GENERATION VALIDATION STEP         */
  'FILEDEF LNKSTMT DISK NCPINCL TEXT A'
/* TEMPORARY WORK FILE USED BY THE TABLE ASSEMBLIES                   */
  'FILEDEF SYSUT1 DISK SYSUT1 TEMP A4 (BLOCK 4000'
/* RUN THE NDF STEP                                                   */
  'ICNRTNDF'
  SELECT
    WHEN (RC = 0) THEN
      DO
/* PUT TEXT OUTPUT FROM TABLE ASSEMBLIES INTO A SIMULATED PDS         */
        'TXTLIB GEN OBJ TABLE1 TABLE2'
        IF RC ¬ = 0 THEN
          DO
            SAY "CANNOT FIND TABLE1 TEXT OR TABLE2 TEXT"
            EXIT 99
          END
/* ERASE TEMPORARY FILES                                              */
        'ERASE SYSUT1 TEMP A'
        'ERASE TABLE1 SOURCE'
        'ERASE TABLE2 SOURCE'
        'ERASE TABLE1 TEXT'
        'ERASE TABLE2 TEXT'
;
        'ESTATE' OBJECT 'TXTLIB *'    /* SEE IF OBJLIB EXISTS         */
        IF RC ¬ = 0 THEN
          SAY "ERROR IN ACCESSING" OBJECT "TXTLIB"
;
/* FILEDEFS FOR THE LINK EDIT STEP                                    */
        'FILEDEF SYSUT1 CLEAR'
        'FILEDEF' OBJECT 'DISK' OBJECT 'TXTLIB *'
/* NCP/EP TABLE TEXT                                                  */
        'FILEDEF SYSPUNCH DISK OBJ TXTLIB A'
```

This example is continued on the next page.

```
/* NAME OF OUTPUT LIBRARY FOR THE LOAD MODULE                          */
        'FILEDEF SYSLMOD DISK' GEN_FN 'LOADLIB  A'
/* RUN LINKAGE EDITOR                                                  */
        'LKED NCPINCL (MAP NCAL NOTERM LET LIST ALIGN2'
        EXIT RC
      END
;
/* FOR TABLE 1 ERROR COPY TABLE 1 LISTING FROM TAPE                    */
    WHEN (RC = 10) THEN
      DO
        'FILEDEF TBL1LIST CLEAR'
        'FILEDEF TBL1LIST TAP1 SL 1 ( BLKSIZE 7260 LRECL 121 RECFM FB'
        'FILEDEF OUTFILE DISK TABLE1 LISTING A (LRECL 121'
        'MOVEFILE TBL1LIST OUTFILE'
        EXIT 10
      END
;
/* FOR TABLE 2 ERROR COPY TABLE 2 LISTING FROM TAPE                    */
    WHEN (RC = 100) THEN
      DO
        'FILEDEF TBL2LIST CLEAR'
        'FILEDEF TBL2LIST TAP1 SL 2 ( BLKSIZE 7260 LRECL 121 RECFM FB'
        'FILEDEF OUTFILE DISK TABLE2 LISTING A  (LRECL 121'
        'MOVEFILE TBL2LIST OUTFILE'
        EXIT 100
      END
;
    OTHERWISE
      DO
        SAY "***ERROR IN EXECUTING NDF***"
        EXIT RC
      END
  END                                  /* END SELECT                 */
```

Example for an NCP/PEP Generation with User-Written Code Using the NDF Standard Attachment Facility

This example shows an EXEC for generating user-written code and NCP using the NDF standard attachment facility. For more information about running this type of generation, see page 4-12. Run this type of user-written code generation **only** if you are using SSP V3R2 to generate NCP V4R2 or the NCP Subset.

To run an NCP/PEP generation with user-written code using the NDF standard attachment facility:

- You must supply a user-written generation application.
- You must code the USERGEN operand on the OPTIONS definition statement as the first executable statement in your generation definition.
- You can choose to code the NEWDEFN operand on the OPTIONS definition statement as the first executable statement in your generation definition if you want NDF to create a new generation definition.
- You must modify the EXEC for a standard NCP/PEP generation as shown below.

Following are examples of the GLOBAL and FILEDEF commands used to include user-written generation load modules in a standard NCP/PEP generation.

```
/* EXAMPLE OF GLOBAL COMMAND TO TO IDENTIFY THE LIBRARY CONTAINING  */
/* A USER WRITTEN GENERATION LOAD MODULE                            */
  'GLOBAL LOADLIB USERLIB'

/* EXAMPLE OF THE FILE DEFINITION FOR NEWDEFN                       */
  'FILEDEF NEWDEFN DISK' GEN_FN 'NEWDEFN' GEN_FM

/* EXAMPLE OF THE FILE DEFINITIONS FOR THE SYSLIB CHAIN FOR A       */
/* NCP/PEP GENERATION WITH USER CODE                                */
  'FILEDEF SYSLIB DISK' MACRO 'MACLIB *'
  'FILEDEF SYSLIB DISK USER1 MACLIB * (CONCAT'
  'FILEDEF SYSLIB DISK USER2 MACLIB * (CONCAT'
  'GLOBAL MACLIB' MACRO 'USER1 USER2'

/* EXAMPLE OF THE FILE DEFINITIONS FOR USER OBJECT LIBRARIES FOR    */
/* THE LINK EDIT OF AN NCP/PEP LOAD MODULE WITH USER CODE           */
  'FILEDEF USER1 DISK USER1 TXTLIB *'
       .          .          .
       .          .          .
       .          .          .
  'FILEDEF USER1 DISK USERN TXTLIB *'
```

Example for an NCP/PEP Generation with User-Written Code Without Using the NDF Standard Attachment Facility

This example shows an EXEC for generating user-written code and NCP without using the NDF standard attachment facility. For more information about running this type of generation, see page 4-13. Run this type of user-written code generation if you are generating any valid version and release of NCP using SSP Version 3. This includes NCP V4R2 and the NCP Subset.

Before you generate, ensure that you:

- Assemble the user-written routines and code the link-edit statements for the routines
- Code the appropriate operands on the GENEND definition statement for your user-written routines
- Place the members with SRCLO or SRCHI code in a macro library (SYSLIB) that is available to NDF
- Place all members that contain INCLUDE or ORDER link-edit control statements in a macro library in the NDF SYSLIB chain
- Place all macros in the NDF SYSLIB chain.

The generation validation phase of NDF reads the link-edit control statements and write them into the same file as the standard NCP link-edit control statements.

Following are examples of GLOBAL and FILEDEF commands used to include the SYSLIB chain and the link-edit ULIB statement in a standard NCP/PEP generation.

```
/* EXAMPLE OF THE FILE DEFINITIONS FOR THE SYSLIB CHAIN FOR A       */
/* NCP/PEP GENERATION WITH USER CODE                                */
  'FILEDEF SYSLIB DISK' MACRO 'MACLIB *'
  'FILEDEF SYSLIB DISK USER1 MACLIB * (CONCAT'
  'FILEDEF SYSLIB DISK USER2 MACLIB * (CONCAT'
  'GLOBAL MACLIB' MACRO 'USER1 USER2'

/* EXAMPLE OF THE FILE DEFINITIONS FOR USER OBJECT LIBRARIES FOR    */
/* THE LINK EDIT OF AN NCP/PEP LOAD MODULE WITH USER CODE           */
/*                                                                  */
/* LIBRARY FOR BLOCK HANDLER AND USER-WRITTEN CODE MODULES          */
  'FILEDEF ULIB DISK USER1 TXTLIB *'
/* LIBRARIES FOR USER-WRITTEN CODE MODULES                          */
  'FILEDEF USER1 DISK USER1 TXTLIB *'
       .         .         .
       .         .         .
       .         .         .
  'FILEDEF USER1 DISK USERN TXTLIB *'
```

Example for a Dynamic Reconfiguration Generation

Use the text file from a dynamic reconfiguration generation to modify an NCP that is already running in a communication controller. Ensure that the original NCP was coded to allow dynamic reconfiguration. The dynamic reconfiguration generation produces a text file that is used by one of the access methods to modify the NCP.

A dynamic reconfiguration generation requires one table assembly and no link edit.

Following is an example of an EXEC for a dynamic reconfiguration generation:

```
/* EXAMPLE OF AN EXEC TO RUN A DYNAMIC RECONFIGURATION GENERATION   */
;
/* COPYRIGHT=NONE                                                   */
;
/* YOU MUST SUPPLY THE FILENAME AND FILETYPE OF THE INPUT GENERATION*/
/* DEFINITION ON THE COMMAND LINE WHEN INVOKING THE EXEC.           */
/* YOU MUST ALSO SUPPLY THE MODEL NUMBER OF THE CONTROLLER.         */
/*                                                                  */
/* YOU ARE ALLOWED TO SPECIFY THE FILEMODE OF YOUR INPUT GENERATION */
/* (WHICH DEFAULTS TO '*'), VERSION OF YOUR GENERATION (WHICH       */
/* DEFAULTS TO 'V3'), AND TEST (WHICH DEFAULTS TO 'T=NO').          */
/* THESE PARAMETERS ARE OPTIONAL.
;
/* CORRECT FORM FOR INVOKING THE EXEC:                              */
/*     VMDR  FN=GEN_FN,FT=GEN_FT,FM=GEN_FM,V=VERSION,M=MODEL,T=NO   */
/*       -GEN_FN, GEN_FT, GEN_FM, VERSION, AND MODEL ARE VARIABLES  */
/*        WHICH YOU SUPPLY ACCORDING TO YOUR GENERATION              */
/*       -YOU MAY ALSO CODE 'T=YES' FOR A RUN ON TEST NCP LIBRARIES */
/*       -ORDER OF PARAMETERS IS NOT IMPORTANT                      */
/*       -SPACES MAY BE USED INSTEAD OF COMMAS                      */
/*       -FOR NCP SUBSET, CODE V=V4S                                */
;
;
/* THE ASSIGNMENT OF THE MACRO LIBRARY NAME IS DERIVED FROM THE     */
/* PARAMETERS PASSED ON THE COMMAND LINE; IT MAY RESIDE ON ANY      */
/* ACCESSED DISK.                                                   */
/*                                                                  */
/* THE MACRO LIBRARY NAME IS REPRESENTED BY THE VARIABLE "MACRO".   */
;
/*******************************************************************/
/* THE FOLLOWING TABLE WILL AID YOU IN WHICH MACRO LIBRARY          */
/* CORRESPONDS TO A PARTICULAR MODEL AND VERSION.  IF YOUR LIBRARY  */
/* NAME DOES NOT AGREE WITH THIS TABLE, YOU WILL NEED TO SUBSTITUTE */
/* YOUR LIBRARY NAME FOR THE STANDARD LIBRARY NAME WHERE APPROPRIATE*/
/* YOU CAN USE THE "ALL" COMMAND ON THE FOLLOWING STRINGS TO FIND   */
/* THE LINES TO CHANGE FOR A PARTICULAR VERSION AND MODEL.          */
/*                                                                  */
```

This example is continued on the next page.

```
/* 'T=YES' IS ALLOWED FOR TESTING TO FACILITATE LIBRARY MAINTENANCE.*/
/* (YOU MUST GO TO THE APPROPRIATE PLACE AND KEY IN YOUR TEST-LIB   */
/*  NAMES TO USE T=YES.)                                            */
/* T=YES WILL RUN WITH ANY MODEL AND VERSION (YOU MUST CODE MODEL). */
/*                                                                  */
/*     ALLTAG1 - TEST                                               */
/*     ALLTAG2 - V3 & 3705                                          */
/*     ALLTAG3 - V3 & 3725                                          */
/*     ALLTAG4 - V4R2 & 3725 OR 3720                                */
/*     ALLTAG5 - V4 SUBSET & 3720                                   */
/*                                                                  */
/*                          M O D E L                               */
/*                                                                  */
/*                3705          3725          3720                */
/*              ------------------------------------                */
/*     V3      |  MAC3705   |  MAC3725   |  NOT        |           */
/*  V          |            |            |  SUPPORTED  |           */
/*  E          |------------------------------------|              */
-/&isR  V4     |  NOT       |  NOT       |  NOT        |           */
/*  S          |  SUPPORTED |  SUPPORTED |  SUPPORTED  |           */
/*  I          |------------------------------------|              */
/*  O  V4R2    |  NOT       |  MAC3725   |  MAC3725    |           */
/*  N          |  SUPPORTED |            |             |           */
/*             |------------------------------------|              */
/*     V4      |  NOT       |  NOT       |  MAC3725    |           */
/*     SUBSET  |  SUPPORTED |  SUPPORTED |             |           */
/*              ------------------------------------                */
/********************************************************************/
;
  ADDRESS COMMAND                       /* ENSURE CP/CMS ENVIRONMENT  */
  TRACE N
  GEN_FN=""
  GEN_FT=""                             /* INITIALIZE STRING VARIABLES*/
  GEN_FM=""
  VERSION=""
  MODEL=""
  T="NO"
  ARG REST                              /* GET PARAMETERS FROM COMMAND*/
                                        /* LINE                       */
  REST=TRANSLATE(REST,' ',',')          /* GET RID OF COMMAS          */
  COUNT=WORDS(REST)
  LPCNT=1
```

This example is continued on the next page.

```
  DO WHILE LPCNT<=COUNT                  /* LOOP THROUGH ONCE FOR EACH */
    TEMP=WORD(REST,LPCNT)                /* WORD IN THE STRING         */
    PARSE VALUE TEMP WITH FRONT '=' BACK
    SELECT
      WHEN (ABBREV(TEMP,'FN')) THEN
        GEN_FN=BACK
      WHEN (ABBREV(TEMP,'FT')) THEN      /* SET APPROPRIATE VARIABLE   */
        GEN_FT=BACK                      /* ACCORDING TO THE ASSIGNMENT*/
      WHEN (ABBREV(TEMP,'FM')) THEN      /* MADE ON THE COMMAND LINE   */
        GEN_FM=BACK
      WHEN (ABBREV(TEMP,'V')) THEN
        VERSION=BACK
      WHEN (ABBREV(TEMP,'M')) THEN
        MODEL=BACK
      WHEN (ABBREV(TEMP,'T')) THEN
        T=BACK
      OTHERWISE
        SAY TEMP" IS NOT VALID, IGNORED"
    END                                  /* END SELECT                 */
    LPCNT=LPCNT+1
  END                                    /* END DO                     */
  IF GEN_FM="" THEN                      /* DEFAULT FILETYPE TO "*" IF */
    GEN_FM="*"                           /* NOT CODED                  */
;
;
/* SEE IF GEN_FN, GEN_FT, AND MODEL WERE PASSED ON COMMAND LINE. IF */
/* GEN NAME OR MODEL WERE NOT SPECIFIED, GIVE CORRECT FORM & EXIT.  */
;
  IF (GEN_FN="")|(GEN_FT="")|(MODEL="") THEN
    DO
      SAY "CORRECT FORM:"
      SAY ""
      SAY "VMDR  FN=GEN_FN,FT=GEN_FT,FM=GEN_FM,V=VERSION,M=MODEL,T=NO"
      SAY ""
      SAY "      -GEN_FN, GEN_FT, GEN_FM, VERSION, MODEL ARE VARIABLES"
      SAY "       WHICH YOU SUPPLY ACCORDING TO YOUR GENERATION"
      SAY "      -YOU MAY CODE 'T=YES' FOR A RUN ON TEST NCP LIBRARIES"
      SAY "      -ORDER OF PARAMETERS IS NOT IMPORTANT"
      SAY "      -SPACES MAY BE USED INSTEAD OF COMMAS"
      SAY "      -FOR NCP SUBSET, CODE V=V4S"
      SAY "      -IF OMITTED, DEFAULTS ARE:"
      SAY "         FM=*"
      SAY "         V=V3"
      SAY "         T=NO"
      SAY "      -GEN_FN, GEN_FT, AND MODEL ARE REQUIRED"
      EXIT
    END
;
;
  'STATE' GEN_FN GEN_FT GEN_FM         /* SEE IF GEN EXISTS ON DISK  */
  IF RC ¬ = 0 THEN
    DO
      SAY GEN_FN GEN_FT GEN_FM "DOES NOT EXIST"
      EXIT RC                            /* EXIT IF GEN DOESN'T EXIST  */
    END
;
;
```

This example is continued on the next page.

```
/* THIS STRUCTURE VALIDATES VERSION AND MODEL AS TAKEN FROM THE     */
/* COMMAND LINE AND SETS "MACRO" ACCORDINGLY.                       */
/*                                                                  */
/* NOTE:  IF YOU HAVE CHANGED A LIBRARY NAME, OR YOUR LIBRARY NAME  */
/* DOES NOT AGREE WITH THE TABLE ABOVE, CHANGE THE ASSIGNMENT       */
/* STATEMENT OF "MACRO" TO REFLECT YOUR LIBRARY NAMES IN THE        */
/* APPROPRIATE PLACE IN THIS STRUCTURE (ACCORDING TO YOUR VERSION   */
/* AND MODEL).                                                      */
;
;
  IF (VERSION="") THEN                  /* VERSION DEFAULTS TO V3     */
    DO                                  /* IF NOT CODED               */
      SAY "DEFAULTING TO VERSION = V3"
      VERSION="V3"
    END
  SELECT
    WHEN (T="YES") THEN
        MACRO=MACXXXX                   /* FOR TEST            ALLTAG1*/
    WHEN (VERSION="V3")&(MODEL="3705") THEN
        MACRO=MAC3705                   /* FOR V3 & 3705       ALLTAG2*/
    WHEN (VERSION="V3")&(MODEL="3725") THEN
        MACRO=MAC3725                   /* FOR V3 & 3725       ALLTAG3*/
    WHEN (VERSION="V4R2")&((MODEL="3725")|(MODEL="3720")) THEN
        MACRO=MAC3725                   /* FOR V4R2&3725|3720  ALLTAG4*/
    WHEN (VERSION="V4S")&(MODEL="3720") THEN
        MACRO=MAC3725                   /* FOR V4 SUBSET       ALLTAG5*/
    OTHERWISE
      DO
        SAY "VERSION = "VERSION" IS NOT VALID WITH MODEL = "MODEL
        EXIT
      END
  END                                   /* END SELECT                 */
;
;
  'ESTATE' MACRO 'MACLIB *'             /* SEE IF MACLIB EXISTS       */
  IF RC ¬ = 0 THEN
      SAY "ERROR IN ACCESSING" MACRO "MACLIB"
;
;
/* CLEAR OLD FILE DEFINITIONS                                       */
  'FILEDEF * CLEAR'
/* WORKING SPILL FILE                                               */
/* THE DBWORKFL IS NEEDED ONLY WHEN THERE IS NOT ENOUGH VIRTUAL     */
/* MEMORY TO HOLD ALL OF NDF'S WORK DATA OR IF USERGEN IS SPECIFIED.*/
/*'FILEDEF DBWORKFL DISK  DBWORKFL FILE A ( XTENT 40'              */
/* MACRO LIBRARIES USED IN THE TABLE ASSEMBLY PHASE OF NDF          */
  'FILEDEF SYSLIB DISK' MACRO 'MACLIB *'
  'GLOBAL MACLIB' MACRO
/* INPUT FILE WITH NCP/EP GENERATION STATEMENTS                     */
  'FILEDEF GENDECK DISK' GEN_FN GEN_FT GEN_FM
/* GENERATION VALIDATION STEP OUTPUT                                */
  'FILEDEF SYSPRINT DISK' GEN_FN 'LISTING A'
/* NDF SUMMARY LISTING                                              */
  'FILEDEF PRINTER TERM'
/* SOURCE FOR TABLE 1 ASSEMBLY - OUTPUT FROM GENERATION VALIDATION  */
  'FILEDEF TBL1SRCE DISK TABLE1 SOURCE A'
/* LISTING FROM THE TABLE1 ASSEMBLY                                 */
  'FILEDEF TBL1LIST DISK TABLE1 LISTING A'
/* TEXT OUTPUT FROM THE TABLE1 ASSEMBLY                             */
  'FILEDEF TBL1OBJ DISK TABLE1 TEXT A'
/* TEMPORARY WORK FILE USED BY THE TABLE ASSEMBLY                   */
  'FILEDEF SYSUT1 DISK SYSUT1 TEMP A4 (BLOCK 4000'
/* RUN THE NDF STEP                                                 */
  'ICNRTNDF'
  EXIT RC
```

Chapter 6. Loading the Program Under VM/SP

After generating an NCP load module, the last step in producing an operating NCP is to load the load module into the communication controller in which it is to reside. You can load your NCP into a channel-attached controller in two ways. You can use the loader utility provided by the Advanced Communications Function for System Support Programs (ACF/SSP or SSP), or you can use a loader facility provided by an access method. If you are loading your NCP into a link-attached controller, you must use the access method loader facility. This manual tells you only how to use the SSP loader utility to load a channel-attached controller. For information on how to use the access method loader facility to load both a channel-attached and link-attached controller, see the appropriate *ACF/VTAM Operation* manual and the appropriate *ACF/TCAM Base Installation Guide.*

A controller module disables all channel adapters except the one over which the load operation takes place. Upon completion of its initialization phase, the NCP enables any additional channel adapters specified as ACTIVE in the NCPCA operand of the BUILD definition statement.

Note: You must manually disable any channel adapter connected to a non-operational host before starting the load process.

Successful completion of the load process is indicated to you by a write-to-operator message. A separate message is issued for each successfully loaded controller. Syntax errors or permanent I/O errors occurring during loading are indicated to you by messages sent to the message file.

A virtual machine can be used to load any IBM 3725, 3720, or 3705 Communication Controller for which a real device block has been generated. The load operation only requires that the controllers's power be on and that the controller be attached to the virtual machine. If you are loading your NCP into an IBM 3725 or 3720 Communication Controller, an additional requirement for the load operation is that the controller's maintenance and operation subsystem (MOSS) be active.

If you are loading your NCP into the IBM 3705 Communications Controller, before the loader utility loads the NCP into the controller, it can load an optional diagnostic routine called the initial test routine. If the initial test routine detects no malfunctions, the loader utility loads the NCP into the controller. If the initial test routine detects trouble, it stops, and the loader utility issues error message IFL004I indicating that fact. The loader utility then loads the remaining controllers, if any, specified in the loader step.

Loading and running of the initial test routine are optional, but are recommended because the routine can detect conditions that can cause later failure of the NCP. The routine is run unless you specify its omission in the LOAD control statement.

Note: If the controller power has been turned off since the last time the initial test routine was run, run the routine again before reloading the controller. This ensures that correct parity is set in the controller storage.

The Loader Utility

This section discusses the loader utility in VM/SP systems. It covers the following:

- Host processor and communication controller requirements
- Input that you supply to the loader utility
- Output from the loader utility.

Host Processor and Communication Controller Requirements

The load module requires a 16K-byte section of user virtual storage.

No work files are required to run the loader utility.

The controller module of the loader utility can be run in any channel-attached communication controller. Before it can be loaded, the controller must:

- Have its power on
- Be identified to the VM/SP system under which the loader utility is running
- Be free to be allocated to the loader
- Not be in program-stop condition
- Have the channel, over which the load is to take place, enabled.

Note: After the loader utility has been started, you should not cancel the load until after normal completion.

The loader utility consists of the load module IFLOADRN, the text files IFLLD1P1, IFLLD1P2, IFLLD2P1, and IFLLD2P2.

For the IBM 3705 Communications Controller, the loader utility also contains one load library, SSPLIB, for the diagnostic routine.

Input to the Loader Utility

The input to the loader utility consists of two files. One is the CMS file (input file) that contains the NCP load module that you load into the controller. The other contains the LOAD control statement specifying the names of the NCP load modules and the controller into which each of these is to be loaded.

If you are loading your NCP into the IBM 3705 Communications Controller, the input to the loader utility consists of an additional file. This third file contains the initial test routine (consisting of modules IFL3705A, IFL3705B, IFL3705D, and IFL3705E), to be loaded into the controller before the NCP load module. This file is optional.

Output from the Loader Utility

The loader utility produces one output listing: SYSPRINT. This listing contains completion or error messages produced by the loader utility. See the *ACF/NCP-SSP Messages and Codes* manual for a description of the messages issued by the loader utility.

Controlling the Loader Utility

This section discusses the VM/SP commands and the utility control statement that you supply to the loader utility. It also presents examples of the VM/SP commands and utility control statements.

VM/SP Commands

To run the loader utility, a number of file definitions must be established. Then the command IFLOADRN must be issued to call the nonrelocatable module generated for the loader utility. The file definition statements you need for calling the loader utility are:

`FILEDEF`	`SYSPRINT`	Specifies the output listing file.
`FILEDEF`	`SYSUT1`	Specifies the input file containing the NCP load module.
`FILEDEF`	`SYSUT3`	Only for the IBM 3705 Communications Controller. Specifies the input file containing the initial test routine load module; not required if you specify `DIAG=NO` in the LOAD statement.
`FILEDEF`	`SYSIN`	Specifies the file (input stream) containing the LOAD control statement.

GLOBAL	LOADLIB	Only for the IBM 3705 Communications Controller. Specifies the LOAD library containing the initial test routine.
IFLOADRN		Specifies the name of the nonrelocatable module generated for the loader utility.

Note: To ensure that the previously defined FILEDEFs are not in effect, clear all file definitions by issuing the command FILEDEF * CLEAR *before* issuing the file definitions for the loader utility.

Utility Control Statement

Only one control statement, the LOAD statement, is needed for the loader utility. It specifies:

- Which member of the input file contains the NCP load module to be loaded
- Which controller you will load
- Whether you will run the initial test routine, if you are loading your NCP into an IBM 3705 Communications Controller.

The following conventions are used in the description of the LOAD statement:

- Capital letters represent values you code directly, without change.
- Small letters represent parameters for which you must supply a value.
- Braces { } indicate that you must choose from the enclosed items.
- "Or" signs | indicate that you can choose between coding various operands
- An underlined value represents the default value of the operand. That is, the loader utility uses that value if you omit the operand.
- Brackets [ ] enclose operands or symbols that are either optional or conditional.

The format of the LOAD statement is:

Name	Operation	Operands
	LOAD	LOADMOD=member name, UNIT=cuu|3725=cuu|3705=cuu [,DIAG={Y6}] {Y8} {NO}

LOADMOD=member name

Specifies which member of the load library indicated by SYSUT1 contains the desired NCP load module. The member must be in standard VM/SP load module form.

UNIT=cuu

For SSP V3R2, specifies the cuu address, which is the virtual subchannel address at which the IBM 3725, 3720, or 3705 Communication Controller is defined.

3725=cuu

Specifies the cuu address, which is the virtual subchannel address at which the IBM 3725 Communications Controller is defined.

3705=cuu

Specifies the cuu address, which is the virtual subchannel address at which the IBM 3705 Communications Controller is defined.

[,DIAG={Y6}]
{Y8}
{NO}

Specifies whether the loader utility is to load the initial test routine into an IBM 3705 Communications Controller.

- **Y6** is the default and specifies that the routine is to be loaded into a controller without extended addressing. A 3705-II having 64K bytes or less of storage uses 16-bit storage addresses and, therefore, does not require extended addressing.
- **Y8** specifies that the routine is to be loaded into a controller with extended addressing. A 3705-II having more than 64K bytes of storage requires extended storage addressing.
- **NO** specifies that the initial routine is not to be loaded at all.

Examples of VM/SP Commands and Utility Control Statements

Example 1: (Using SSP V3R2)

Assume that you want to load an NCP load module named NCP2, residing on a CMS disk, into an IBM 3725 or 3720 Communication Controller whose unit address is 030. The EXEC you would use to accomplish this loading is:

```
/* Exec to call the loader for an IBM 3725 */
/* or 3720 Communication Controller        */

address command

'filedef * clear'
'filedef sysut1 disk ncp2 loadlib a'
'filedef sysprint terminal'
'filedef sysin disk ncp2 card a'
'ifloadrn'

exit
```

where disk file NCP2 CARD A contains:

```
LOAD LOADMOD=NCP2,UNIT=030
```

Example 2: (Using SSP V3R2)

Assume that you want to load an NCP load module named NCP1, residing on a CMS disk, into an IBM 3705 Communications Controller whose unit address is 030. The EXEC you would use to accomplish this loading is:

```
/* Exec to call the loader for an IBM 3705 */
/* Communications Controller               */

address command

'filedef * clear'
'filedef sysut1 disk ncp1 loadlib a'
'filedef sysprint terminal'
'filedef sysut3 disk ssplib loadlib a'
'filedef sysin disk ncp1 card a'
'global loadlib ssplib'
'ifloadrn'

exit
```

where disk file NCP1 CARD A contains:

```
LOAD LOADMOD=NCP1,UNIT=030,DIAG=Y8
```

Note: This example shows that the initial test routine is to be loaded and run before the NCP load module is loaded. If you did not want the initial test routine to be loaded and run, you would include DIAG=NO on the LOAD statement, and would omit the SYSUT3 FILEDEF and GLOBAL LOADLIB statements.

Example 3: (Using SSP V3R1)

Assume that you want to load an NCP load module named NCP2, residing on a CMS disk, into an IBM 3725 Communication Controller whose unit address is 030. The EXEC you would use to accomplish this loading is:

```
/* Exec to call the loader for an IBM 3725 */
/* Communication Controller                */

address command

'filedef * clear'
'filedef sysut1 disk ncp2 loadlib a'
'filedef sysprint terminal'
'filedef sysin disk ncp2 card a'
'ifloadrn'

exit
```

where disk file NCP2 CARD A contains:

```
LOAD LOADMOD=NCP2,3725=030
```

Example 4: (Using SSP V3R1)

Assume that you want to load an NCP load module named NCP1, residing on a CMS disk, into an IBM 3705 Communications Controller whose unit address is 030. The EXEC you would use to accomplish this loading is:

```
/* Exec to call the loader for an IBM 3705 */
/* Communications Controller               */

address command

'filedef * clear'
'filedef sysut1 disk ncp1 loadlib a'
'filedef sysprint terminal'
'filedef sysut3 disk ssplib loadlib a'
'filedef sysin disk ncp1 card a'
'global loadlib ssplib'
'ifloadrn'

exit
```

where disk file NCP1 CARD A contains:

```
LOAD LOADMOD=NCP1,3705=030,DIAG=Y8
```

Note: This example shows that the initial test routine is to be loaded and run before the NCP load module is loaded. If you did not want the initial test routine to be loaded and run, you would include DIAG=NO on the LOAD statement, and would omit the SYSUT3 FILEDEF and GLOBAL LOADLIB statements.

3

Generating and Loading Under VSE

Chapter 7 Generating the Program Under VSE
Chapter 8 Examples of Job Control Lanquage for Generation Under VSE
Chapter 9 Loading the Program Under VSE

Part Three: Generating and Loading Under VSE

Chapter 7. Generating the Program Under VSE 7-1
Understanding the Generation Procedure 7-2
Controlling the Generation Procedure 7-5
Understanding Listings and Error Messages 7-12

Chapter 8. Examples of Job Control Language for Generation Under VSE 8-1
Example for a FASTRUN Generation 8-2
Example for an NCP/PEP Generation 8-3
Example for an NCP/PEP Generation With User-Written Code or Special IBM Products 8-6
Example for a Dynamic Reconfiguration Generation 8-7

Chapter 9. Loading the Program Under VSE 9-1
The Loader Utility 9-2
Controlling the Loader Utility 9-4

Chapter 7. Generating the Program Under VSE

After you install your NCP and SSP product from the tape, and after you define the NCP configuration, the next step in producing an operating NCP is to generate the program. The generation procedure can be run in any host processor; it does not have to be run in the host processor that will load the NCP phases. However, for the NCP Subset, the generation procedure must be run in the host processor with the same operating system of the host that will load the NCP phases.

This chapter contains information about generating an NCP under the VSE operating system. It consists of three parts:

- Understanding the Generation Procedure
- Controlling the Generation Procedure
- Understanding Listings and Error Messages.

SSP Version 3 includes the NCP/EP Definition Facility (NDF), a program used in generating NCP and/or EP phases. NDF can be used in performing several different tasks. These are:

- FASTRUN validation of an NCP/PEP generation definition
- Generation of NCP/PEP phases
- Generation of NCP/PEP Subset phases
- Generation of NCP/PEP phases with user-written code or special IBM products
- Generation of a text file for Dynamic Reconfiguration.

Before running NDF you must supply Job Control Language (JCL) to control the generation procedure. NDF does not create any JCL for you. This chapter later discusses how to control the generation procedure, and in Chapter 8, "Examples of Job Control Language for Generation Under VSE" this manual supplies examples of JCL for generation.

Understanding the Generation Procedure

Generating an NCP with NDF under the VSE operating system is a six step process. For a diagram of the input NDF accepts and the output it produces under the VSE operating system, see Figure 7-1 on page 7-3.

Generation Steps

Step One: In the first step, the generation validation step, NDF does the following:

- Reads your generation definition file and validates the definition statements and operands coded in the generation definition
- Generates assembler language source code for the resources coded in the generation definition
- Creates link-edit control statements that will be used later to link control block objects with preassembled NCP code objects to generate NCP phases.

Step one produces two outputs: a listing and a file with input for the following table assembly steps.

Step Two, Step Three, and *Step Four:* These three steps are all table assemblies. The assemblies transform a source code specification for NCP and/or EP control blocks into object code representation. A single output file is used for all three assemblies, so that only one file is used as input for the VSE LIBRARIAN.

The Table 1 and Table 2 assemblies require NCP and/or EP macros, because most of the control blocks are specified by macro calls. The Table 3 assembly requires no macros.

Step Five: This step calls the LIBRARIAN to catalog the output object modules and link-edit statements from the three table assemblies into the appropriate sublibrary.

Step Six: The last step, the linkage editor step, links the control block objects with the appropriate preassembled NCP code objects and generates the NCP phases.

Note: If you want to do a FASTRUN generation to validate your generation definition without creating control blocks, **do not** specify in your JCL that you want the table assemblies and the link edit to be run. If you want to do a generation for a dynamic reconfiguration, specify in your JCL that you want **only** one table assembly to be run and that you **do not** want the link edit to be run.

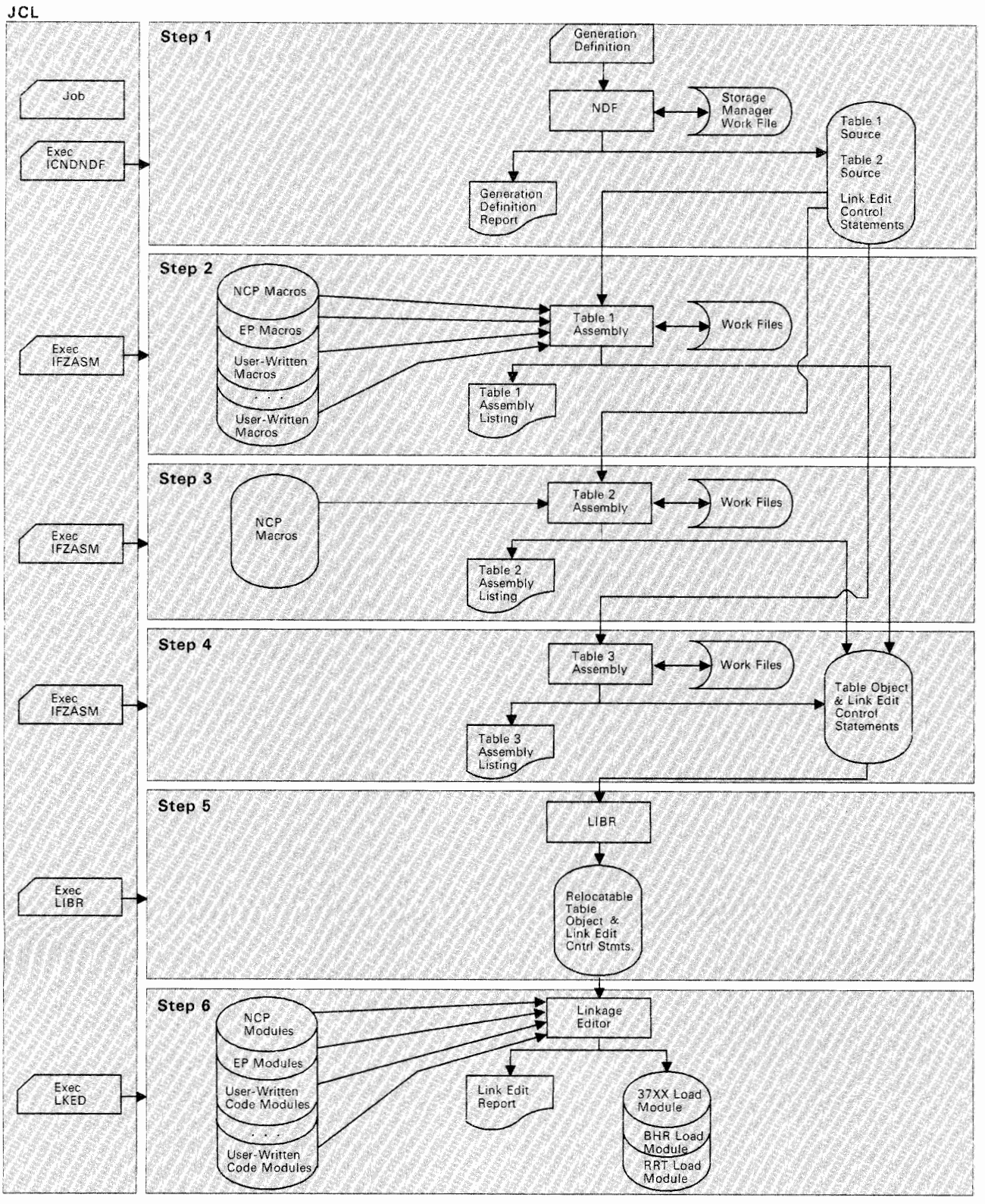


Figure 7-1. The Generation Procedure under VSE

DASD Work Space Requirements

NDF uses a storage manager to organize its work space during NDF generation validation. Whenever possible, storage manager data is kept in virtual storage. The storage manager buffers, Virtual Storage Access Method (VSAM) space, and buffers for the non-VSAM files make up the GETVIS region. A fixed 64K (K = 1024) bytes of virtual memory is reserved for VSAM, and most of the rest of the space is taken by the storage manager for buffers. If data overflows these buffers, this extra data is written into a work file.

NDF jobs require a fixed quantity of about 200K bytes for work space and an additional 48 bytes for each resource specified in the generation definition. Generally enough GETVIS space is available to hold all the storage manager data.

However, if you need additional work space, you need to define a work file, called DBWRKFL. To define DBWRKFL, use ACCESS method services to define a cluster for a relative record file of 4096 byte records. Define this cluster prior to the NDF job that uses DBWRKFL. The following is an example of an IDCAMS job to define such a cluster:

```
// JOB DEFCLUST
*
*  COPYRIGHT=NONE
*
*  EXAMPLE OF A JOB TO DEFINE A VSAM CLUSTER FOR THE DBWRKFL
*
// LIBDEF PHASE,SEARCH=NCPLIB.PR$AM2
// EXEC IDCAMS,SIZE=AUTO
   DEFINE CLUSTER                    -
        (  NAME(VSAM.WORK)           -
           VOL(DT9354)               -
           CISZ(4608)               -
           NUMBERED                  -
           RECORDSIZE(4096 4096)     -
           TRACKS(10 5))             -
       DATA                          -
        (NAME(VSAM.WORK.DATA)        -
         FILE(DBWRKFL))
/&
```

If you define DBWRKFL, generally less than one cylinder of disk space defined for VSAM space should be adequate.

Performance Considerations

NDF requires one megabyte of real storage for optimal performance. If the available real storage drops below one megabyte, paging during the generation validation phase significantly degrades performance.

During the generation procedure, intermediate files are used to transfer NDF's output to IFZASM and to transfer IFZASM's output to the LIBRARIAN. These files are written as SYSIPT. With FBA devices, these files are blocked. For this reason, you can achieve some reduction in generation time by using FBA devices for these two files.

Controlling the Generation Procedure

You control the generation procedure through the job control language (JCL) that you code and through the definition statements and operands that you coded in the generation definition.

This section explains the different types of generations you can run. It discusses some of the definition statements and operands you can code in your generation definition. It also discusses the parameters you need to code and the files you need to specify in your JCL. For examples of JCL for generating, see page 8-1.

Specifying Files Used by NDF

This section contains the names of the files, which you specify in your JCL, that are used by NDF. Below are listed the names and descriptions of these files.

DTF name	Description
IJSYSIN	Specifies the input file for NDF. This file contains the NCP and/or EP generation definition. IJSYSIN is also the DTF name for the input files for the IFZASM assembler and for the LIBRARIAN step that catalogs the NCP table objects into relocatable members.
DBWRKFL	Specifies NDF's work file. This file can be used to temporarily store internal data in 4K byte records. This file is a VSAM relative record file. If you need this file, ensure that it has been defined with IDCAMS before NDF is started.
IJSYSPH	Specifies the file into which NDF writes the input for all three assemblies. The inputs for different assemblies are separated by a "/*" statement. This file is read as input by the assembler for each of the assemblies. IJSYSPH also identifies the file used by the assembler for text output. The outputs from all three assemblies are written sequentially to one file. This file is then referred to as IJSYSIN by the LIBRARIAN.

Specifying Parameters for NDF

This section contains descriptions of the optional parameters for NDF that you can specify in your JCL. When specifying more than one parameter in the parameter field, you must separate the parameters with a comma.

The LINECNT Parameter

Use the LINECNT parameter to specify the number of lines on each page of the generation validation listing and the table assembly listing. The valid range for this parameter is from 10 to 99. The default value for the validation listing and for the assembly listing is 60. If you specify a value for LINECNT, this value is used in all listings.

The following is an example of the specification of LINECNT in the JCL:

```
//EXEC PGM=ICNDNDF,SIZE=AUTO,PARM='LINECNT=40'
```

The FASTRUN Parameter

Use the FASTRUN parameter to check for errors before running a complete generation. A FASTRUN generation checks your generation definition for syntax and definition errors without creating control blocks or link-edit control statements.

Note: Using the FASTRUN parameter is the same as coding FASTRUN=ON on the OPTIONS definition statement as the first executable statement in your generation definition.

The following is an example of the specification of FASTRUN in the JCL:

```
//EXEC PGM=ICNDNDF,SIZE=AUTO,PARM='FASTRUN=ON'
```

Naming Resources

Avoid using the following prefixes when naming resources, because they are used as control block identifiers and can cause duplicate labels that result in an error message from the assembler.

$	CB	DVI	LCB	NET	RH *	TRT
AAB	CBB	DVQ	LCC	NIB	RN *	TVS
ABN	CCB	ECB	LCI	NIX	RU *	UAC
ACB	CDS	ECD	LCP	NLB	R *	UCD
ACT	CER	ECL	LCS	NLX	RMB	UIB
ACU	CGP	EML	LCW	NPB	RST	UIC
AEB	CHC	EPI	LGT	NPF	RTR	ULVSGN
AST	CHV	EQB	LKB	NQB	RVT	UNA
ATB	CIE	ERB	LKC	NQE	SCB	USC
ATP	CM	ERX	LNV	NSQ	SGE	U1
AXB	COE	FCT	LPB	NVT	SGT	VAT
BC	CPI	FLB	LRC	NVX	SID	VIT
BCU	CPN	FMT	LTC	OLL	SIT	VLB
BER	CPT	FVT	LTR	OLT	SMB	VR
BGS	CRB	GCB	LTS	PAB	SMM	VST
BH	CRP	GPT	LTV	PAD	SNP	VTS
BHD	CTP	GRW	LU	PCB	SOT	VVT
BHR	CUB	GVT	LX	PIU	SST	WCB
BHS	CY	HWE	L1B	PMF	STE	WRP
BLU	CX	HWX	L4B	PSA	STQ	WU
BOQ	DAE	IB	MBF	PSB	SUT	X
BPB	DDB	ICE	MBX	PST	SVT	XDA
BST	DIA	ICW	MCT	PUV	SXB	XDB
BTT	DPT	IDD	MDR	QAB	SYS	XDH
BTU	DQB	IDE	MIB	QCB	TCB	XID
BUE	DRS	IDL	MIF	RAT	TET	
CAB	DRX	IOB	MLT	RCB	TGB	
CAI	DSP	IRN	MMV	RCQ	TH *	
CAR	DTG	LAA	MSC	RCV	TIM	
CAT	DVB	LAB	MTF	RG	TND	

* Followed by any numerical digit 0 through 9.

Figure 7-2. Label Prefixes to Avoid. The above list should not be considered all-inclusive. Avoid names that are similar to control block acronyms.

Defining Virtual Storage

You can control virtual storage available to NDF for work space by specifying SIZE=AUTO on the EXEC statement. Specifying SIZE=AUTO allows the system to determine the amount of virtual storage available.

About 1.5 megabytes of virtual storage is required to run NDF. If storage is exceeded, increase virtual storage or define the DBWRKFL file.

Below is an example of the SIZE=AUTO specification in the JCL:

```
// EXEC ICNDNDF,SIZE=AUTO
```

Naming Phases

In addition to creating NCP phases, NDF also produces a resource resolution table (RRT) phase and, if you have coded any block handling routines, a block-handler set resolution table (BHR) phase. The RRT and BHR phases contain information required by the access method.

The NCP, RRT, and BHR phases are placed in a sublibrary. The sublibrary is determined by the ACCESS librarian command.

Use the NEWNAME operand on the BUILD definition statement to designate the names for the BHR, RRT, and NCP phases. NDF appends a "B" to the NEWNAME value to specify the name for BHR phases, and NDF appends an "R" to the NEWNAME value to specify the name for RRT phases.

For information about the NEWNAME operand on the BUILD definition statement, see the *ACF/NCP-SSP Resource Definition Guide*. For information on how to code this operand, see the *ACF/NCP-SSP Resource Definition Reference*.

If you are generating your NCP for the IBM 3725 or 3720 Communication Controller, the link edit always produces six phases. If you are generating your NCP for the IBM 3705 Communications Controller, the number of phases depends on whether you included user-written code or block-handlers, and whether the storage of the user code is high or low. The following table illustrates that the number of phases is dependent on the relationship between storage of user-written code, operands specified, and block handlers coded. This relationship is valid only for the IBM 3705 Communications Controller.

If you included	and specified on the GENEND definition statement	the storage of user code is	and the NCP has this number of phases
no user code			1
user code	SRCLO and/or INCLO	low	2
user code	SRCHI and/or INCHI	high	2
user block handlers		high	2
user code	SRCLO and/or INCLO & SRCHI and/or INCHI	low & high	3
user code & user block handlers	SRCLO, INCHI, and/or INCLO	low & high	3

Figure 7-3. Determining the Number of Phases for an IBM 3705 Communications Controller

When more than one phase is present, the first phase is named with the value specified for the NEWNAME operand. The other phase names are derived from this value. All phase names, except the first, are 8 bytes long. The 8 bytes are made up of the NEWNAME value with zeroes concatenated to 7 bytes. The eighth byte contains a suffix starting at 2 and incremented by 1 for each phase.

Example 1: If NEWNAME=NCPA for an NCP generated for the IBM 3725 or 3720 Communication Controller, the phase names are NCPA, NCPA0002, NCPA0003, NCPA0004, NCPA0005, and NCPA0006.

Example 2: If NEWNAME=NCPA for an NCP generated for the IBM 3705 Communications Controller and you specified INCLO and SRCHI on the GENEND definition statement, then the NCP has three phases named NCPA, NCPA0002, and NCPA0003.

Controlling Succeeding Generation Steps

In the VSE environment, a system return code is produced that can be used in determining whether or not succeeding job steps will be run. The examples of JCL in Chapter 8, "Examples of Job Control Language for Generation Under VSE" use condition code checking before initiating succeeding job steps.

Running a FASTRUN Generation

Do a FASTRUN generation to check for errors before running a complete generation. A FASTRUN generation checks your generation definition for syntax and definition errors without creating control blocks or link-edit control statements.

To do a FASTRUN generation, code FASTRUN=ON on the OPTIONS definition statement as the first executable statement in your generation definition, or code FASTRUN=ON as a parameter in your JCL when calling NDF. Ensure that your JCL does not call the linkage editor. If the linkage-editor step is present, an error condition will result. Also, do not define the NCP macro library, since NDF does not run table assemblies for a FASTRUN generation. However, if you include user-written code in the generation definition, define the macro library that contains user-written link-edit control statements.

For an example of the JCL for a FASTRUN generation, see page 8-2.

Running a Standard NCP/PEP Generation

To run a standard NCP/PEP generation, you supply your generation definition as input and specify the various input and output files in your JCL.

Ensure that you allow the LENAME operand on the BUILD definition statement to be defaulted to INLINKED, or if you code a value for this operand, ensure that you use the same value name on the INCLUDE statement in the link-edit step of your JCL.

For an example of JCL for running a standard NCP/PEP generation, see "Example for an NCP/PEP Generation" on page 8-3.

Running an NCP/PEP Generation with User-Written Code or Special IBM Products

When you include user-written code or special IBM products (for example, X.25 NCP Packet Switching Interface or Network Routing Facility) in an NCP or PEP generation, you may need to modify the basic JCL. To run this type of generation you must code link-edit statements and CSECTS for your user routine. You must also identify the location of link-edit statements by coding certain operands on the GENEND definition statement.

Before you generate, ensure that you:

- Run any job required to generate SRCLO or SRCHI code.
- Catalog the members with SRCLO or SRCHI code as members of type "A".

- Edit and catalog any macros called by the SRCLO or SRCHI code as members of type "F".
- Catalog link-edit control statements and preassembled object code as members of type "OBJ".
- Add sublibraries that contain source code or edited macros to the LIBDEF chain for SOURCE. (Establish this LIBDEF chain before IFZASM is called for the table assemblies.)
- Add sublibraries that contain link-edit control statements to the LIBDEF chain for OBJ. (Establish this LIBDEF before you call the linkage editor.)

For an example of the JCL for generating user-written code or special IBM products, see page 8-6.

Running a Dynamic Reconfiguration Generation

Use the text file from a dynamic reconfiguration generation to modify an NCP that is already running in a communication controller. Ensure that the original NCP was coded to allow dynamic reconfiguration. The dynamic reconfiguration generation produces a text file that is used by one of the access methods to modify the NCP.

A dynamic reconfiguration generation requires one table assembly and no link edit.

For an example of JCL for a dynamic reconfiguration generation, see page 8-7.

Understanding Listings and Error Messages

During generation validation, NDF creates a report that contains:

- The input statements interspersed with informational and error messages
- A resource name and network address cross reference (only in valid runs)
- An error message summary
- A return code for the generation validation step.

The input for all three table assembly steps is contained in one output file. This file is the SYSPCH file during the NDF generation validation step and the SYSIPT file during the table assemblies.

If any errors occur in generation validation, NDF notes these errors through diagnostic messages in the report. If these messages show serious errors (severity 4 or greater), NDF ends output of control-block source and link-edit control statements, but continues to validate the input definition statements. In this case, you must correct the errors and rerun. If no serious errors occur in NDF or the table assemblies, NDF runs to completion, runs the table assemblies, executes the link edit, and produces phases.

Note: Other procedures, such as VTAM and CRP, require as input the same definition statements coded to generate the NCP, plus several additional operands and statements specific to each procedure. The additional operands and statements are identified in the NCP/SSP Resource Definition Reference. Although you can place these additional operands and statements in the NCP generation definition either before or after you generate the NCP, it is strongly recommended that you add them before, because executing the different procedures with different inputs can create errors.

NDF does not check the accuracy of the values coded for procedures, other than the NCP generation, that appear in the NCP generation input. It checks only the keyword part for proper spelling. In the same way, these other procedures do not check the validity of the NCP definition statements and operands. Because of this, NDF requires that the NCP generation have no errors of severity 4 or greater.

Sample Generation Report

This section contains an example of an NDF generation report. Comments about the report are cited by numbers enclosed in angle brackets, for example <3>, and are not a part of the actual report. You can find the comments corresponding to these numbers following the report.
Ellipses (. . .) in the report indicate that parts of the report were deleted for this example.

```
1                                  2                                  3
ACF SSP V3          05/09/84 09:52:13    DEFINITION SPECIFICATION                    PAGE 5
 4
LINE #                STATEMENT
   177  *          FULL LINE COMMENT  5
   178  G1481      GROUP CUTOFF=1,           ONE SUB-BLOCK ACCEPTED  6          X
   179                   DIAL=NO,           NON-SWITCHED LINES                 X
   180                   LNCTL=BSC,         BSC LINE CONTROL                   X
   181                   NPACOLL=YES,       COLLECT NPA ON THIS GROUP          X
   182                   PU=YES,            (V) VTAM.  FOR CROSS DOMAIN        X
                  . . .
                     D             WAKDLAY=2,2
                     D             SYNDLAY=1,0
                     D             TTDCNT=15
                     D             WACKCNT=15
   201  L14023     LINE  ADDRESS=(G23),      LINE ADDRESS ON 3705               X
   202                   CODE=EBCDIC,       EBCDIC 3270'S ONLY                 X
   203                   CUTYPE=3271,       3271'S DEFINED                     X
   204                   DUPLEX=FULL,       FULL DUPLEX FACILITY               X
   205                   ETRATIO=30,        ERROR-TO-TRANSMISSION RATIO=3% NPDA X
   206                   INTPRI=1,          INTERRUPT PRIORITY IS 1            X
   207                   LPDATS=YES,        LINE PROBLEM DETERMINATION SUPPORTEDX
   208                   NEGPOLP=.2,        PAUSE .2 SEC AFTER NEGATIVE RESPONSEX
   209                   NEWSYNC=YES,       CONTROLLER SUPPLIES NEW-SYNC SIGNAL X
               . . .
                     G             TYPE=NCP
*ERROR*    ICN001I  08  ADDRESS=(G23) INVALID, REQUIRED  8
                     G             CLOCKNG=EXT
                     D             DATRATE=LOW
                     D             SPDSEL=NO  7
                     D             PROMPT=YES
                     D             DIALALT=NONE
                     G             TRANSFR=3
               . . .
   259  B14023A    CLUSTER  CRITSIT=YES,     SEND CLOSE-DOWN MESSAGE          X00000400
   260                   NPACOLL=YES,       DATA CAN BE COLLECTED FOR NPA    X00000500
   261                   GPOLL=40407F7F,    GENERAL POLL ADDRESS             X00000600
   262                   XMITLIM=1,         SEND OR RECEIVE ONE TRANSMISSION X00000700
   263                   ISTATUS=ACTIVE     (V) VTAM                          00000800
                     L             CUTYPE=3271
                     D             CDATA=NO
                     D             INHIBIT=NONE
                     D             LGRAPHS(1)=REJECT
                     D             LGRAPHS(2)=REJECT
                     D             FEATURE=NOGPLKUP
                     D             ITBMODE(1)=NO
                     D             ITBMODE(2)=NO
                     D             BHSET=NONE
               . . .
ACF SSP V3          05/09/84 13:51:30    LABEL CROSS REFERENCE                       PAGE 41

                    LABEL CROSS REFERENCE -- SORTED BY LABEL NAME  9

LABEL          LINE      SA  ELEM        LABEL          LINE      SA  ELEM        LABEL          LINE      SA  ELEM
               . . .
G1481          178                       T14020BB       545       0E  001B        T14022G8       915       0E  004D
G14S1          362                       T14020BC       549       0E  001C        T14022G9       916       0E  004E
               . . .
L14022         828       0E  0044        T14020B8       533       0E  0018        T14023A7       305       0E  0009
L14023         201       0E  0001        T14020B9       537       0E  0019        T14023A8       315       0E  000A
               . . .
ACF SSP V3          05/09/84 13:51:30    LABEL CROSS REFERENCE                       PAGE 42

                    LABEL CROSS REFERENCE -- SORTED BY NETWORK ADDRESS
                                                                10
SA  ELEM    LINE    LABEL              SA  ELEM    LINE    LABEL              SA  ELEM    LINE    LABEL
               . . .
0E  0001    201     L14023             0E  0038    747     T14020F8           0E  006E    90      DRPOOLLU
0E  0002    259     B14023A            0E  0039    749     T14020F9           0E  006F    6       NCPBUILD
0E  0003    269     T14023A1           0E  003A    767     P14020G            0E  0070    6       NCPBUILD
0E  0004    275     T14023A2           0E  003B    785     T14020G1           0E  0071    6       NCPBUILD
               . . .
ACF SSP V3          05/09/84 09:51:30    ERROR SUMMARY                               PAGE 41

TOTAL MESSAGES      INFO      WARNING      ERROR      SEVERE      FATAL
      1              0          0           1          0           0

RETURN CODE IS 8


MESSAGES APPEAR AFTER THE LINES NUMBERED:  11

209

REGENERATION REQUIRED
               . . .
ACF SSP V3          05/09/84 09:52:13    NDP RETURN CODE SUMMARY


GEN DECK PROCESSOR          8
TABLE ONE BUILD        NOT RUN
TABLE TWO BUILD        NOT RUN
    NDF OVERALL              1
```

Figure 7-4. Sample from an NDF Generation Report.

Comments

1 This portion of the header line on the report page contains the SSP version description.

2 This portion of the header line on the report page contains the date and time of the NDF run. This is the same date and time recorded in the date/time control block in the NCP/EP EP phases and printed in the formatted portion of the NCP/EP dump.

3 This portion of the header line on the report page identifies the report section. It has one of the following values: "DEFINITION SPECIFICATION," "LABEL CROSS REFERENCE," or "ERROR SUMMARY."

4 The line number column contains the line numbers of the generation definition listing.

5 NDF accepts full-line assembler comments.

6 This line contains a partial-line assembler comment.

7 This part of the report contains information describing operands that are defaulted or inherited.

A code prefixed to the associated message indicates operands that are defaulted or operands using values from previous definition statements. These prefixes are:

G Keyword inherited from GROUP
L Keyword inherited from LINE
T Keyword inherited from TERMINAL
C Keyword inherited from CLUSTER
P Keyword inherited from PU
D Keyword that has been defaulted

8 Errors are indicated by an appropriate error number followed by a severity code and error message text. A severity code of 4 or more require corrections to the generation definition before you can generate phases. A severity code of less than 4 informs you that NDF has taken corrective action and does not require regeneration.

9 The first label cross reference is sorted by label name. All user-coded labels appear in this list. If applicable, a network address is printed. Not all labels have associated network addresses. This section is not printed when severity codes of 4 or more exist. It is included in this sample for illustrative purposes only.

10 The second label cross reference is sorted by network address. Labels with no associated network addresses are omitted. This section is not printed when severity codes of 4 or more exist.

11 The error summary section contains an error count and a list of the line numbers immediately preceding error messages. If more than one error message immediately follows a given line, the line number is printed only once. If only informational messages follow a given line, an asterisk is be printed next to the line number.

Chapter 8. Examples of Job Control Language for Generation Under VSE

This chapter contains examples of job control language for generating your NCP under the VSE operating system. These examples are supplied here for easy reference and can also be found on the SSP tape sent from IBM Software Distribution. However, the example of JCL for generating user-written code cannot be found on the SSP tape. Before using these examples, ensure that they match your operational environment.

This chapter includes examples of job control language for the following types of generations:

- A FASTRUN generation
- An NCP/PEP generation
- An NCP/PEP generation with user-written code or special IBM products
- A dynamic reconfiguration generation.

Example for a FASTRUN Generation

A FASTRUN generation validates your generation definition without creating control blocks or link-edit control statements. Do a FASTRUN generation to check for errors before running a complete generation.

To do a FASTRUN generation, code FASTRUN=ON on the OPTIONS definition statement as the first definition statement in your generation definition, or code FASTRUN=ON as a parameter in your JCL when calling NDF. This example uses the FASTRUN parameter coded in the JCL.

Ensure that your JCL **does not** call the table assemblies or the linkage editor.

Following is an example of JCL for a FASTRUN generation:

```
// JOB NDF
*
*  COPYRIGHT=NONE
*
*  THE SSP MEMBERS REQUIRED TO CARRY OUT AN NCP GENERATION ARE
*  ASSUMED TO RESIDE IN THE SSPLIB SUBLIBRARY.
*
*  A LIBDEF STATEMENT IS NEEDED TO INDICATE THE SUBLIBRARY
*  WHERE NDF RESIDES.  THE SUBLIBRARY WHERE VSAM PHASES
*  RESIDE MUST ALSO BE PART OF THE SEARCH CHAIN IF THE
*  WORKFILE (DBWRKFL) IS TO BE USED.
// LIBDEF PHASE,SEARCH=(NCPLIB.SSPLIB,NCPLIB.PR$AM2)
*
*  THIS EXAMPLE ASSUMES THAT THE GENERATION DECK IS INCLUDED IN
*  LINE.  IF IT IS ON DISK, A SERIES OF JOB CONTROL STATEMENTS SIMILAR
*  TO THE FOLLOWING ARE NEEDED.
*  // DLBL IJSYSIN,'SAMPLE GENERATION'
*  // EXTENT SYSIPT
*  // ASSGN SYSIPT,DISK,PERM,VOL=DSKID,SHR
*
*  THE LISTING IS SENT TO A PRINTER
// ASSGN SYSLST,00E,PERM
*
*  THE STORAGE MANAGER WORK FILE IS NEEDED ONLY WHEN THERE IS NOT
*  ENOUGH VIRTUAL MEMORY TO HOLD ALL OF NDF'S WORK DATA OR IF
*  USERGEN IS SPECIFIED.
*  // DLBL DBWRKFL,'VSAM.WORK',0,VSAM
*
*  NDF IS EXECUTED WITH SIZE=AUTO SO THAT ANY EXTRA VIRTUAL
*  MEMORY WILL BE AVAILABLE IN THE GETVIS AREA
// EXEC ICNDNDF,SIZE=AUTO,PARM='LINECNT=55,FASTRUN=ON'
*
*  PLACE YOUR GENERATION INPUT FILE HERE
*
/*
/&
```

Example for an NCP/PEP Generation

This section gives an example of JCL for an NCP/PEP generation. This JCL assumes that the LENAME operand on the BUILD definition statement was defaulted to INLINKED. If you coded a value for this operand, ensure that you use the same value name on the INCLUDE statement in the link-edit step in your JCL. This example includes conditional JCL to stop the job when one of the steps fails.

This example shows the JCL used to generate NCP phases for the IBM 3725 Communication Controller. For an IBM 3705 or 3720 Communication Controller:

- The JCL is slightly different.
- The NCP macro library used for the NDF job step may be different.
- The ddname for the library of preassembled NCP object modules in the link-edit step may be different.

These differences are explained in the text included in the following example.

```
// JOB NDF
*
* COPYRIGHT=NONE
*
*********************************************************************
* THE FOLLOWING TABLE WILL AID YOU IN WHICH MACRO AND OBJECT         *
* LIBRARIES CORRESPOND TO A PARTICULAR MODEL AND VERSION.  IF YOU    *
* CHANGED YOUR LIBRARY NAMES WHEN THE PRODUCT WAS INSTALLED, YOU     *
* MAY WANT TO UPDATE THIS TABLE.  BE SURE TO CHECK THE "LIBDEF"      *
* STATEMENTS THAT DEFINE THESE LIBRARIES.                            *
*                                                                    *
*                          M O D E L                                 *
*                                                                    *
*                3705          3725          3720                  *
*              ---------------------------------------               *
*     V3      |  MAC3705   |  MAC3725   |  NOT       |              *
*  V          |  OBJ3705   |  OBJ3725   |  SUPPORTED |              *
*  E          |-------------------------------------|               *
*  R  V4      |  NOT       |  MAC3725   |  MAC3725   |              *
*  S          |  SUPPORTED |  OBJ3725   |  OBJ3725   |              *
*  I          |-------------------------------------|               *
*  O  V4      |  NOT       |  NOT       |  MAC3725   |              *
*  N  SUBSET  |  SUPPORTED |  SUPPORTED |  OBJ3725   |              *
*             |-------------------------------------|               *
*     V4R2    |  NOT       |  NOT       |  NOT       |              *
*             |  SUPPORTED |  SUPPORTED |  SUPPORTED |              *
*              ---------------------------------------               *
*********************************************************************
```

This example is continued on the next page.

```
*
*   THIS EXAMPLE ASSUMES THAT ALL PROGRAMS RELATED TO THE NCP ARE
*   KEPT IN A SINGLE LIBRARY, NCPLIB.  THE SSP MEMBERS REQUIRED TO
*   CARRY OUT AN NCP GENERATION ARE ASSUMED TO RESIDE IN THE
*   SSPLIB SUBLIBRARY.  THE NCP LOAD MODULES ARE PLACED IN THE
*   NCPLOAD SUBLIBRARY.
*
*   A LIBDEF STATEMENT IS NEEDED TO INDICATE THE SUBLIBRARY
*   WHERE NDF AND IFZASM RESIDE.  THE SUBLIBRARY WHERE VSAM PHASES
*   RESIDE MUST ALSO BE PART OF THE SEARCH CHAIN IF THE
*   WORKFILE (DBWRKFL) IS TO BE USED.
// LIBDEF PHASE,SEARCH=(NCPLIB.SSPLIB,NCPLIB.PR$AM2)
*
*   THIS EXAMPLE ASSUMES THAT THE GENERATION DECK IS INCLUDED IN
*   LINE.  IF IT IS ON DISK, A SERIES OF JOB CONTROL STATEMENTS SIMILAR
*   TO THE FOLLOWING ARE NEEDED.
*   // DLBL IJSYSIN,'SAMPLE GENERATION'
*   // EXTENT SYSIPT
*   // ASSGN SYSIPT,DISK,PERM,VOL=DSKID,SHR
*
*   THE SYSPCH FILE IS USED AS THE PUNCH CODE OUTPUT FILE BY NDF.
*   THIS FILE WILL BE USED AS THE INPUT FILE FOR THE TABLE ASSEMBLIES.
// DLBL IJSYSPH,'TMP FILE',0001,SD
// EXTENT SYSPCH,YYYYYY,,,1501,1200
// ASSGN SYSPCH,DISK,PERM,VOL=YYYYYY,SHR
*
*   THE LISTING IS SENT TO A PRINTER
// ASSGN SYSLST,00E,PERM
*
*   THE STORAGE MANAGER WORK FILE IS NEEDED ONLY WHEN THERE IS NOT
*   ENOUGH VIRTUAL MEMORY TO HOLD ALL OF NDF'S WORK DATA OR IF
*   USERGEN IS SPECIFIED.
*   // DLBL DBWRKFL,'VSAM.WORK',0,VSAM
*
*   NDF IS EXECUTED WITH SIZE=AUTO SO THAT ANY EXTRA VIRTUAL
*   MEMORY WILL BE AVAILABLE IN THE GETVIS AREA
// EXEC ICNDNDF,SIZE=AUTO,PARM='LINECNT=55'
*
*   PLACE YOUR GENERATION INPUT FILE HERE
*
/*
// IF $RC GE 4 THEN
// GOTO CLSPCH
*
*   THE CLOSE COMMAND IS NEEDED BEFORE REASSIGNMENT OF SYSTEM
*   LOGICAL UNITS.  IF THE NDF INPUT FILE IS ON DISK A CLOSE IS
*   ALSO NEEDED FOR SYSIPT.
CLOSE SYSPCH,UA
*
```

This example is continued on the next page.

```
*  DEFINE THE INPUT FILE FOR ALL THREE ASSEMBLIES
// DLBL IJSYSIN,'TMP FILE'
// EXTENT SYSIPT
// ASSGN SYSIPT,DISK,PERM,VOL=YYYYYY,SHR
*
*  DEFINE THE PUNCH CODE OUTPUT FILE FOR ALL THREE ASSEMBLIES
// DLBL IJSYSPH,'TABLE TEXT',0001,SD
// EXTENT SYSPCH,YYYYYY,,,2701,500
// ASSGN SYSPCH,DISK,PERM,VOL=YYYYYY,SHR
*
*  DEFINE THE NCP MACRO LIBRARY
// LIBDEF SOURCE,SEARCH=(NCPLIB.MAC3725)
*
*  THE DECK OPTION IS REQUIRED TO GET THE PROPER OUTPUT FROM THE
*  TABLE ASSEMBLIES.  THE SXREF OPTION WILL PROVIDE AN ADEQUATE XREF
// OPTION DECK,SXREF
// EXEC IFZASM
// IF $RC GT 4 THEN
// GOTO CLSBOTH
// EXEC IFZASM
// IF $RC GT 4 THEN
// GOTO CLSBOTH
// EXEC IFZASM
// IF $RC GT 0 THEN
// GOTO CLSBOTH
CLOSE SYSIPT,UA
CLOSE SYSPCH,00D
*  THE LIBRARIAN MUST NOW BE USED TO CATALOG THE TABLE TEXT FILES
*  INTO OBJ MEMBERS OF THE LIBRARY
*
*  THE PUNCH CODE OUTPUT FILE FROM THE TABLE ASSEMBLIES SERVES
*  AS THE INPUT FILE FOR THE LIBRARIAN JOB
// DLBL IJSYSIN,'TABLE TEXT'
// EXTENT SYSIPT
// ASSGN SYSIPT,DISK,PERM,VOL=YYYYYY,SHR
// EXEC LIBR,PARM='ACCESS SUBLIB=NCPLIB.NCPLOAD'
// IF $RC GT 0 THEN
// GOTO CLSIPT
*
*  A LIBDEF STATEMENT MUST BE USED TO DEFINE THE SEARCH CHAIN
*  FOR THE INPUT TO THE LINK EDIT STEP.
*  DEFINE THE NCP OBJECT LIBRARY
// LIBDEF OBJ,SEARCH=(NCPLIB.OBJ3725,NCPLIB.NCPLOAD)
*
*  A LIBDEF STATEMENT MUST ALSO BE USED TO DEFINE THE SUBLIBRARY
*  WHERE THE LINK EDIT OUTPUT PHASES WILL BE CATALOGED
// LIBDEF PHASE,CATALOG=NCPLIB.NCPLOAD
*
// OPTION CATAL
   ACTION MAP,NOAUTO
   INCLUDE INLINKED
// EXEC LNKEDT
// GOTO CLSIPT
*  NEED TO CLOSE DIFFERENT COMBINATIONS OF SYSIPT AND SYSPCH
*  DEPENDING ON WHERE THE JOB TERMINATED
/. CLSPCH
CLOSE SYSPCH,00D
// GOTO FINISH
/. CLSBOTH
CLOSE SYSPCH,00D
/. CLSIPT
CLOSE SYSIPT,00C
/. FINISH
/&
```

Example for an NCP/PEP Generation With User-Written Code or Special IBM Products

When you include user-written code or special IBM products (for example, X.25 NCP Packet Switching Interface and Network Routing Facility) in an NCP generation, you may need to modify the basic JCL. Before you run NDF, ensure that you:

- Run any job required to generate SRCLO or SRCHI code.
- Catalog the members with SRCLO or SRCHI code members of type "A".
- Edit and catalog any macros called by the SRCLO or SRCHI code as members of type "F".
- Catalog link-edit control statements and preassembled object code as members of type "OBJ".
- Add sublibraries that contain source code or edited macros to the LIBDEF chain for SOURCE. (Establish this LIBDEF chain before IFZASM is called for the table assemblies.)
- Add sublibraries that contain link-edit control statements to the LIBDEF chain for OBJ. (Establish this LIBDEF before you call the linkage editor.)

Following are examples of how to specify the table assembly search chain and the link-edit search chain in the JCL for a standard NCP generation.

```
*
*   COPYRIGHT=NONE
*
*   LIBDEF FOR THE TABLE ASSEMBLY SEARCH CHAIN
*
*   USER SOURCE CODE HAS BEEN CATALOGED IN D TYPE MEMBERS OF
*   THE USERMAC SUBLIB
*
*   EDITED MACROS CALLED FROM THE USER SOURCE CODE HAVE BEEN
*   CATALOGED IN F TYPE MEMBERS OF THE USERMAC SUBLIBRARY
*
// LIBDEF SOURCE,SEARCH=(NCPLIB.MAC3725,NCPLIB.USERMAC)
*
*
*   LIBDEF FOR THE LINK EDIT SEARCH CHAIN
*
*   MEMBERS WITH INCLUDE STATEMENTS FOR USER CODE HAVE BEEN CATALOGED
*   IN OBJ TYPE MEMBERS OF THE USEROBJ SUBLIBRARY.  THE NAMES OF
*   THESE MEMBERS WERE CODED FOR ONE OF THE "INCxxx" KEYWORDS ON
*   THE GENEND STATEMENT
*
*   THE INCLUDED PREASSEMBLED USER OBJECT MODULES HAVE ALSO BEEN
*   CATALOGED IN OBJ TYPE MEMBERS IN THE USEROBJ SUBLIBRARY
*
// LIBDEF OBJ,SEARCH=(NCPLIB.OBJ3725,NCPLIB.USEROBJ)
```

Example for a Dynamic Reconfiguration Generation

Use the text file from a dynamic reconfiguration generation to modify an NCP that is already running in a communication controller. Ensure that you coded the original NCP to allow for dynamic reconfiguration. The dynamic reconfiguration generation produces a text file that is used by one of the access methods to modify the NCP.

A dynamic reconfiguration job requires one table assembly and no link edit.

Following is an example of a job for a dynamic reconfiguration generation:

```
// JOB NDF
*
* COPYRIGHT=NONE
*
*********************************************************************
* THE FOLLOWING TABLE WILL AID YOU IN WHICH MACRO LIBRARY            *
* CORRESPONDS TO A PARTICULAR MODEL AND VERSION.  IF YOU CHANGED     *
* YOUR LIBRARY NAME WHEN THE PRODUCT WAS INSTALLED, YOU MAY WANT     *
* TO UPDATE THIS TABLE.  BE SURE TO CHECK THE "LIBDEF" STATEMENT     *
* THAT DEFINES THIS LIBRARY.                                         *
*                                                                    *
*                          M O D E L                                 *
*                                                                    *
*                3705          3725          3720                  *
*              ------------------------------------                  *
*     V3      |  MAC3705   |  MAC3725   |  NOT       |              *
*  V          |            |            |  SUPPORTED |              *
*  E          |------------------------------------|                *
*  R  V4      |  NOT       |  MAC3725   |  MAC3725   |              *
*  S          |  SUPPORTED |            |            |              *
*  I          |------------------------------------|                *
*  O  V4      |  NOT       |  NOT       |  MAC3720   |              *
*  N  SUBSET  |  SUPPORTED |  SUPPORTED |            |              *
*             |------------------------------------|                *
*     V4R2    |  NOT       |  NOT       |  NOT       |              *
*             |  SUPPORTED |  SUPPORTED |  SUPPORTED |              *
*              ------------------------------------                  *
*********************************************************************
*
*  THIS EXAMPLE ASSUMES THAT ALL PROGRAMS RELATED TO THE NCP ARE
*  KEPT IN A SINGLE LIBRARY, NCPLIB.  THE SSP MEMBERS REQUIRED TO
*  CARRY OUT AN NCP GENERATION ARE ASSUMED TO RESIDE IN THE
*  SSPLIB SUBLIBRARY.  THE NCP LOAD MODULES ARE PLACED IN THE
*  NCPLOAD SUBLIBRARY.
*
```

This example is continued on the next page.

```
*  A LIBDEF STATEMENT IS NEEDED TO INDICATE THE SUBLIBRARY
*  WHERE NDF AND IFZASM RESIDE.  THE SUBLIBRARY WHERE VSAM PHASES
*  RESIDE MUST ALSO BE PART OF THE SEARCH CHAIN IF THE
*  WORKFILE (DBWRKFL) IS TO BE USED.
// LIBDEF PHASE,SEARCH=(NCPLIB.SSPLIB,NCPLIB.PR$AM2)
*
*  THIS EXAMPLE ASSUMES THAT THE GENERATION DECK IS INCLUDED IN
*  LINE.  IF IT IS ON DISK, A SERIES OF JOB CONTROL STATEMENTS SIMILAR
*  TO THE FOLLOWING ARE NEEDED.
*  // DLBL IJSYSIN,'SAMPLE GENERATION'
*  // EXTENT SYSIPT
*  // ASSGN SYSIPT,DISK,PERM,VOL=DSKID,SHR
*
*  THE SYSPCH FILE IS USED AS THE PUNCH CODE OUTPUT FILE BY NDF.
*  THIS FILE WILL BE USED AS THE INPUT FILE FOR THE TABLE ASSEMBLY.
// DLBL IJSYSPH,'TMP FILE',0001,SD
// EXTENT SYSPCH,YYYYYY,,,1501,1200
// ASSGN SYSPCH,DISK,PERM,VOL=YYYYYY,SHR
*
*  THE LISTING IS SENT TO A PRINTER
// ASSGN SYSLST,00E,PERM
*
*  THE STORAGE MANAGER WORK FILE IS NEEDED ONLY WHEN THERE IS NOT
*  ENOUGH VIRTUAL MEMORY TO HOLD ALL OF NDF'S WORK DATA OR IF
*  USERGEN IS SPECIFIED.
*  // DLBL DBWRKFL,'VSAM.WORK',0,VSAM
*
*  NDF IS EXECUTED WITH SIZE=AUTO SO THAT ANY EXTRA VIRTUAL
*  MEMORY WILL BE AVAILABLE IN THE GETVIS AREA
// EXEC ICNDNDF,SIZE=AUTO,PARM='LINECNT=55'
*
*  PLACE YOUR DYNAMIC RECONFIGURATON SOURCE HERE
*
/*
// IF $RC GE 4 THEN
// GOTO CLSPCH
*
*  THE CLOSE COMMAND IS NEEDED BEFORE REASSIGNMENT OF SYSTEM
*  LOGICAL UNITS.  IF THE NDF INPUT FILE IS ON DISK, A CLOSE IS
*  ALSO NEEDED FOR SYSIPT.  SYSPCH IS ASSIGNED TO A PUNCH SO
*  THAT THE DR TEXT WILL APPEAR AS PUNCHED OUTPUT.
CLOSE SYSPCH,00D
*
*  DEFINE THE INPUT FILE FOR THE ASSEMBLY.
// DLBL IJSYSIN,'TMP FILE'
// EXTENT SYSIPT
// ASSGN SYSIPT,DISK,PERM,VOL=YYYYYY,SHR
*
*
*  DEFINE THE NCP MACRO LIBRARY
// LIBDEF SOURCE,SEARCH=(NCPLIB.MAC3725)
*
*  THE DECK OPTION IS REQUIRED TO GET THE PROPER OUTPUT FROM THE
*  TABLE ASSEMBLIES.  THE SXREF OPTION WILL PROVIDE AN ADEQUATE XREF
// OPTION DECK,SXREF
// EXEC IFZASM
CLOSE SYSIPT,00C
// GOTO FINISH
/. CLSPCH
CLOSE SYSPCH,00D
/. FINISH
/&
```

Chapter 9. Loading the Program Under VSE

After generating the NCP, the last step in producing an operating NCP is to load the phases into the communication controller in which they are to reside. You can load your NCP into a channel-attached controller in two ways. You can use the loader utility provided by the Advanced Communications Function for System Support Programs (ACF/SSP or SSP), or you can use a loader facility provided by an access method. If you are loading your NCP into a link-attached controller, you must use the access method loader facility. This manual tells you only how to use the SSP loader utility to load a channel-attached controller. For information on how to use the access method loader facility to load both a channel-attached and link-attached controller, see the appropriate *ACF/VTAM Operation* manual and the appropriate *ACF/TCAM Base Installation Guide.*

The SSP loader utility is run as a job or job step under the operating system. A controller module disables all channel adapters except the one over which the load operation takes place. Upon completion of its initialization phase, the NCP enables any additional channel adapters specified as ACTIVE in the NCPCA operand of the BUILD definition statement.

Note: You must manually disable any channel adapter connected to a non-operational host before starting the load process. Successful completion of the load process is indicated to you by a write-to-operator message. A separate message is issued for each successfully loaded controller. Syntax errors or permanent I/O errors occurring during loading are indicated to you by messages sent to the message logical unit.

If you are loading your NCP into the IBM 3705 Communications Controller, before the loader utility loads the NCP into the controller, it can load an optional diagnostic routine called the initial test routine. If the initial test routine detects no malfunctions, the loader utility loads the NCP into the controller. If the initial test routine detects trouble, it stops, and the loader utility issues error message IFL004I indicating that fact. The loader utility then loads the remaining controllers, if any, specified in the loader job. Loading and running of the initial test routine are optional, but are recommended because the routine can detect conditions that can cause later failure of the NCP. The routine is run unless you specify its omission in the LOAD control statement.

Note: If the controller power has been turned off since the last time the initial test routine was run, run the routine again before reloading the controller. This ensures that correct parity is set in the controller storage.

The Loader Utility

This section discusses the loader utility in VSE systems. It covers the following:

- Host processor and communication controller requirements
- Input that you supply to the loader utility
- Output from the loader utility.

Host Processor and Communication Controller Requirements

The loader utility in VSE systems operates in a minimum virtual partition.

No work files are required to run the loader utility.

The controller module of the loader can be run in any channel-attached communication controller. Before it can be loaded, the controller must:

- Have its power on
- Be identified to the operating system under which the loader utility will be run
- Be free to be allocated to the loader job step
- Not be in a program-stop condition
- Have the channel, over which the load is to take place, enabled.

The loader utility consists of the phase IFULOAD.

Input to the Loader Utility

The input to the loader utility consists of two files. One is a DASD file (input file) that contains the NCP phases that you want to load into the controller. The other contains a LOAD statement specifying the name of the NCP phases and the controller into which it is to be loaded.

If you are loading your NCP into the IBM 3705 Communications Controller, the input to the loader utility consists of an additional file. This third file contains the initial test routine (consisting of phases IFU3705D and IFU3705E), to be loaded into the controller before the NCP phases. This file is optional. It can be omitted if you do not want the initial test routine to be run.

Before you load your NCP into the controller, ensure that you punch all the phases onto the disk by using the LIBRARIAN. The following is an example of a job that moves the NCP phases for an IBM 3725 or 3720 Communication Controller from the SSPLIB.GEN sublibrary onto the disk:

```
// JOB      INITTEST
*  DEFINE SYSTEM PUNCH TO DISK LOCATION WHERE PHASES WILL BE STORED
// DLBL     IJSYSPH, other parameters
// EXTENT   SYSPCH, other parameters
*  SPECIFY PUNCH UNIT ADDRESS
// ASSGN    SYSPCH,X'xxx'
*  INVOKE LIBRARIAN TO PUNCH PHASES
// EXEC     LIBR
   ACCESS   S=SSPLIB.GEN
   PUNCH    VSE8.PHASE,VSE80002.PHASE,VSE80003.PHASE,-
            VSE80004.PHASE,VSE80005.PHASE,VSE80006.PHASE
/*
   CLOSE    SYSPCH,X'xxx'
/&
```

If you are loading your NCP into the IBM 3705 Communications Controller, you must also move the phases making up the initial test routine. The following is an example of a job that moves the phases VSENCP1, IFU3705D, and IFU3705E from the SSPLIB.GEN sublibrary onto the disk:

```
// JOB      INITTEST
*  DEFINE SYSTEM PUNCH TO DISK LOCATION WHERE PHASES WILL BE STORED
// DLBL     IJSYSPH, other parameters
// EXTENT   SYSPCH, other parameters
*  SPECIFY PUNCH UNIT ADDRESS
// ASSGN    SYSPCH,X'xxx'
*  INVOKE LIBRARIAN TO PUNCH PHASES
// EXEC     LIBR
   ACCESS   S=SSPLIB.GEN
   PUNCH    VSENCP1.PHASE,IFU3705D.PHASE,IFU3705E.PHASE
/*
   CLOSE    SYSPCH,X'xxx'
/&
```

Output from the Loader Utility

The loader utility produces one output logical unit: the message logical unit SYSLST. This logical unit contains completion or error messages produced by the loader utility. See the *ACF/NCP-SSP Messages and Codes* manual for a description of the messages issued by the loader utility.

Controlling the Loader Utility

This section discusses the job control statements and the utility control statement that you supply to the loader utility. It presents examples of the job and utility control statements. It also explains how to link edit object code into phases.

Job Control Statements

The job control statements you need for calling the loader utility are:

//	JOB	Starts the job.
//	ASSGN	Specifies the unit address of the controller to be loaded. This statement can be omitted if a permanent assignment exists for the controller.
//	DLBL	Defines a sequential file that contains suitably formatted phases.
//	EXTENT	Correlates the DLBL statement to the disk.
//	ASSGN	Assigns the file defined in the previous DLBL and EXTENT statements.
//	LIBDEF	Specifies the location of the loader utility.
//	DLBL	`DIAGFLE,'file-id'` Only for the IBM 3705 Communications Controller. Defines the sequential file that contains the initial test routine; not required if you specify `DIAG=NO` in the LOAD statement.
//	EXTENT	`SYS008,vol.id,1` Only for the IBM 3705 Communications Controller. Correlates the DLBL statement to the disk. Required only if you specify or imply `DIAG=Y6 or Y8` on any LOAD statement.
//	ASSGN	`SYS008,X'ccu'` Only for the IBM 3705 Communications Controller. Assigns the file defined in the previous DLBL and EXTENT statements. Required only if you specify or imply `DIAG=Y6 or Y8` on any LOAD statement.
//	EXEC	Specifies the program name, IFULOAD.

Utility Control Statement

Only one utility control statement is needed to call the loader utility. This is the LOAD statement. It specifies:

- Which member of the input file contains the NCP phases to be loaded
- Which controller you will load
- Whether you want to run the initial test routine, if you are loading your NCP into an IBM 3705 Communications Controller.

The following conventions are used in the description of the LOAD statement:

- Capital letters represent values you code directly, without change.
- Small letters represent parameters for which you must supply a value.
- Braces { } indicate that you must choose from the enclosed items.
- "Or" signs | indicate that you can choose between coding various operands.
- An underlined value represents the default value of the operand. That is, the loader utility uses that value if you omit the operand.
- Brackets [ ] enclose operands or symbols that are either optional or conditional.

The format of the LOAD statement is:

Name	Operation	Operands
	LOAD	LOADMOD=file name, UNIT=SYSxxx|3725=SYSxxx|3705=SYSxxx [,DIAG={Y6}] {Y8} {NO}

`LOADMOD=file name`

Specifies the name of the file that contains the NCP phases. This name must be the same as the file name specified in the DLBL statement.

`UNIT=SYSxxx`

For SSP V3R2, specifies the symbolic name of the IBM 3725, 3720, or 3705 Communication Controller to be loaded.

`3725=SYSxxx`

Specifies the symbolic name of the IBM 3725 Communication Controller to be loaded.

```
3705=SYSxxx
```

Specifies the symbolic name of the IBM 3705 Communications Controller to be loaded.

```
[,DIAG={Y6}]
       {Y8}
       {NO}
```

Specifies whether the loader utility is to load the initial test routine into an IBM 3705 Communications Controller.

- **Y6** is the default and specifies that the routine is to be loaded into a controller without extended addressing. A 3705-II having 64K bytes or less of storage uses 16-bit storage addresses and, therefore, does not require extended addressing.
- **Y8** specifies that the routine is to be loaded into a controller with extended addressing. A 3705-II having more than 64K bytes of storage requires extended storage addressing.
- **NO** specifies that the initial routine is not to be loaded at all.

Examples of Job and Utility Control Statements

Example 1: (Using SSP V3R2)

Assume that you want to load NCP phases named NCP3MOD, residing in a file named NCPFILE, into an IBM 3725 or 3720 Communication Controller whose unit address is 001. The job and utility statements you would use to accomplish this loading are:

```
//     JOB          LOADUNIT
//     ASSGN        SYS007,X'001'
//     DLBL         NCPFILE,'NCP3MOD'
//     EXTENT       SYS005,111111
//     ASSGN        SYS005,X'131'
//     LIBDEF       PHASE,SEARCH=(SSPLIB.LOADER)
//     EXEC         IFULOAD
       LOAD         LOADMOD=NCPFILE,UNIT=SYS007
/*
/&
```

If the NCP phases reside on a fixed block architecture (FBA) device, the file it would reside in would be named FBAFLE1 and the job and utility statements you would use to accomplish the loading would be:

```
//     JOB          LOADUNIT
//     ASSGN        SYS007,X'001'
//     DLBL         FBAFLE1,'NCP3MOD'
//     EXTENT       SYS001,111111
//     ASSGN        SYS001,X'131'
//     LIBDEF       PHASE,SEARCH=(SSPLIB.LOADER)
//     EXEC         IFULOAD
       LOAD         LOADMOD=FBAFLE1,UNIT=SYS007
/*
/&
```

Example 2: (Using SSP V3R2)

Assume that you want to load NCP phases named NCP3MOD, with the initial test routine named INITTEST (residing in a file named DIAGFLE), into an IBM 3705 Communications Controller whose unit address is 001. The job and utility statements you would use to accomplish this loading are:

```
//     JOB            LOAD3705
//     ASSGN          SYS007,X'001'
//     DLBL           DIAGFLE,'INITTEST'
//     EXTENT         SYS008,111111
//     ASSGN          SYS008,X'131'
//     DLBL           NCPFILE,'NCP3MOD'
//     EXTENT         SYS005,111111
//     ASSGN          SYS005,X'131'
//     LIBDEF         PHASE,SEARCH=(SSPLIB.LOADER)
//     EXEC           IFULOAD
       LOAD           LOADMOD=NCPFILE,UNIT=SYS007,DIAG=Y8
/*
/&
```

Example 3: (Using SSP V3R1)

Assume that you want to load NCP phases named NCP3MOD, residing in a file named NCPFILE, into an IBM 3725 Communication Controller whose unit address is 001. The job and utility statements you would use to accomplish this loading are:

```
//     JOB            LOADUNIT
//     ASSGN          SYS007,X'001'
//     DLBL           NCPFILE,'NCP3MOD'
//     EXTENT         SYS005,111111
//     ASSGN          SYS005,X'131'
//     LIBDEF         PHASE,SEARCH=(SSPLIB.LOADER)
//     EXEC           IFULOAD
       LOAD           LOADMOD=NCPFILE,3725=SYS007
/*
/&
```

If the NCP phases reside on a fixed block architecture (FBA) device, the file it would reside in would be named FBAFLE1 and the job and utility statements you would use to accomplish the loading would be:

```
//     JOB            LOADUNIT
//     ASSGN          SYS007,X'001'
//     DLBL           FBAFLE1,'NCP3MOD'
//     EXTENT         SYS001,111111
//     ASSGN          SYS001,X'131'
//     LIBDEF         PHASE,SEARCH=(SSPLIB.LOADER)
//     EXEC           IFULOAD
       LOAD           LOADMOD=FBAFLE1,3725=SYS007
/*
/&
```

Example 4: (Using SSP V3R1)

Assume that you want to load NCP phases named NCP3MOD, with the initial test routine named INITTEST (residing in a file named DIAGFLE), into an IBM 3705 Communications Controller whose unit address is 001. The job and utility statements you would use to accomplish this loading are:

```
//     JOB          LOAD3705
//     ASSGN        SYS007,X'001'
//     DLBL         DIAGFLE,'INITTEST'
//     EXTENT       SYS008,111111
//     ASSGN        SYS008,X'131'
//     DLBL         NCPFILE,'NCP3MOD'
//     EXTENT       SYS005,111111
//     ASSGN        SYS005,X'131'
//     LIBDEF       PHASE,SEARCH=(SSPLIB.LOADER)
//     EXEC         IFULOAD
       LOAD         LOADMOD=NCPFILE,3705=SYS007,DIAG=Y8
/*
/&
```

Link-Editing Object Code into Phases

If the host processor phases of the loader utility are cataloged as object code in the sublibrary, you can use the following control statements to link-edit them into phases in the sublibrary.

```
//     JOB          LINKLOAD
//     OPTION       CATAL
//     LIBDEF       OBJ,SEARCH=(SSPLIB.LOADER)
//     LIBDEF       PHASE,CATALOG=(SSPLIB.LOADER),PERM
       INCLUDE      IFULINK
//     EXEC         LNKEDT
/*
/&
```

In addition to the above JCL, include LIBDEF statements to tell the program where to locate objects, and into which file to place phases.

4

Glossary and Index

Glossary

This glossary defines important NCP and SSP abbreviations and terms. It includes information from the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699. Definitions from the *American National Dictionary for Information Processing* are identified by an asterisk (*). Definitions from draft proposals and working papers under development by the International Standards Organization, Technical Committee 97, Subcommittee 1 are identified by the symbol **(TC97)**. Definitions from the *CCIT Sixth Plenary Assembly Orange Book, Terms and Definitions* and working documents published by the Consultative Committee on International Telegraph and Telephone of the International Telecommunication Union, Geneva, 1980 are preceded by the symbol **(CCITT/ITU)**. Definitions from published sections of the *ISO Vocabulary of Data Processing*, developed by the International Standards Organization, Technical Committee 97, Subcommittee 1 and from published sections of the *ISO Vocabulary of Office Machines*, developed by subcommittees of ISO Technical Committee 95, are preceded by the symbol **(ISO)**.

For abbreviations, the definition usually consists only of the words represented by the letters; for complete definitions, see the entries for the words.

Reference Words Used in the Entries

The following reference words are used in this glossary:

Contrast with. Refers to a term that has an opposed or substantively different meaning.

Deprecated term for. Indicates that the term should not be used. It refers to a preferred term, which is defined.

See. Refers to multiple-word terms that have the same last word.

See also. Refers to related terms that have similar (but not synonymous) meanings.

Synonym for. Appears in the commentary of a less desirable or less specific term and identifies the preferred term that has the same meaning.

Synonymous with. Appears in the commentary of a preferred term and identifies less desirable or less specific terms that have the same meaning.

ACB. (1) In VTAM, application control block. (2) In NCP, adapter control block.

ACB name. (1) The name of an ACB macro instruction. (2) A name specified in the ACBNAME parameter of a VTAM APPL statement. Contrast with *network name.*

accept. For a VTAM application program, to establish a session with a logical unit (LU) in response to a CINIT request from a system services control point (SSCP). The session-initiation request may begin when a terminal user logs on, a VTAM application program issues a macro instruction, or a VTAM operator issues a command. See also *acquire (1).*

access method. A technique for moving data between main storage and input/output devices.

ACF. Advanced Communications Function.

ACF/NCP. Advanced Communications Function for the Network Control Program. Synonym for *NCP.*

ACF/SSP. Advanced Communications Function for the System Support Programs. Synonym for *SSP.*

ACF/TCAM. Advanced Communications Function for the Telecommunications Access Method. Synonym for *TCAM.*

ACF/VTAM. Advanced Communications Function for the Virtual Telecommunications Access Method. Synonym for *VTAM.*

acquire. (1) For a VTAM application program, to initiate and establish a session with another logical unit (LU). The acquire process begins when the application program issues a macro instruction. See also *accept*. (2) To take over resources that were formerly controlled by an access method in another domain, or to resume control of resources that were controlled by this domain but released. Contrast with *release*. See also *resource takeover*.

active. (1) The state a resource is in when it has been activated and is operational. Contrast with *inactive, pending,* and *inoperative*. (2) Pertaining to a major or minor node that has been activated by VTAM. Most resources are activated as part of VTAM start processing or as the result of a VARY ACT command.

adapter control block (ACB). In NCP, a control block that contains line control information and the states of I/O operations for BSC lines, start-stop lines, or SDLC links.

Advanced Communications Function (ACF). A group of IBM program products (principally VTAM, TCAM, NCP, and SSP) that use the concepts of Systems Network Architecture (SNA), including distribution of function and resource sharing.

application control block (ACB). A control block that links an application program to VSAM or VTAM.

available. In VTAM, pertaining to a logical unit that is active, connected, enabled, and not at its session limit.

basic transmission unit (BTU). In SNA, the unit of data and control information passed between path control components. A BTU can consist of one or more path information units (PIUs). See also *blocking of PIUs*.

BIU segment. In SNA, the portion of a basic information unit (BIU) that is contained within a path information unit (PIU). It consists of either a request/response header (RH) followed by all or a portion of a request/response unit (RU), or only a portion of an RU.

blocking of PIUs. In SNA, an optional function of path control that combines multiple path information units (PIUs) into a single basic transmission unit (BTU).

boundary function. In SNA: (1) A capability of a subarea node to provide protocol support for adjacent peripheral nodes, such as: (a) transforming network addresses to local addresses, and vice versa; (b) performing session sequence numbering for low-function peripheral nodes; and (c) providing session-level pacing support. (2) The component that provides these capabilities. See also *path control (PC) network* and *network addressable unit (NAU)*.

BTU. Basic transmission unit.

chain. See *RU chain*.

channel adapter. A communication controller hardware unit used to attach the controller to a System/360 or a System/370 channel.

channel-attached. Pertaining to the attachment of devices directly by data channels (I/O channels) to a host processor. Contrast with *link-attached*. Synonymous with *local-attached*.

CMS. Conversational Monitor System.

command. (1) A request from a terminal for the performance of an operation or the execution of a particular program. (2) In SNA, any field set in the transmission header (TH), request header (RH), and sometimes portions of a request unit (RU), that initiates an action or that begins a protocol; for example: (a) Bind Session (session-control request unit), a command that activates an LU-LU session, (b) the change-direction indicator in the RH of the last RU of a chain, (c) the virtual route reset window indicator in a FID4 transmission header. See also *VTAM operator command*.

communication controller. A type of communication control unit whose operations are controlled by one or more programs stored and executed in the unit; for example, the IBM 3725 Communication Controller. It manages the details of line control and the routing of data through a network.

communication line. Deprecated term for *telecommunication line* and *transmission line*.

configuration. (1) (TC97) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. The term may refer to a hardware or a software configuration. (2) The devices and programs that make up a system, subsystem, or network. (3) In CCP, the arrangement of controllers, lines, and terminals attached to an IBM 3710 Network Controller. Also, the collective set of item definitions that describe such a configuration.

configuration report program (CRP). An SSP utility program that creates a configuration report

listing network resources and resource attributes for networks with NCP, EP, PEP, or VTAM.

connected. In VTAM, pertaining to a physical unit (PU) or logical unit (LU) that has an active physical path to the host processor containing the system services control point (SSCP) that controls the PU or LU.

connection. Synonym for *physical connection.*

control block. (ISO) A storage area used by a computer program to hold control information.

control program (CP). The VM operating system that manages the real processor's resources and is responsible for simulating System/370s for individual users.

control statement. A statement in a command list that controls the processing sequence of the command list or allows the command list to send messages to the operator and receive input from the operator.

Conversational Monitor System (CMS). A VM application program for general interactive time sharing, problem solving, and program development.

CP. Control program.

CRP. Configuration report program.

DASD. Direct access storage device.

data flow control (DFC) layer. In SNA, the layer within a half-session that (1) controls whether the half-session can send, receive, or concurrently send and receive request units (RUs); (2) groups related RUs into RU chains; (3) delimits transactions via the bracket protocol; (4) controls the interlocking of requests and responses in accordance with control modes specified at session activation; (5) generates sequence numbers; and (6) correlates requests and responses.

data link. In SNA, synonym for *link.*

data link control (DLC) layer. In SNA, the layer that consists of the link stations that schedule data transfer over a link between two nodes and perform error control for the link. Examples of data link control are SDLC for serial-by-bit link connection and data link control for the System/370 channel.

definite response (DR). In SNA, a value in the form-of-response-requested field of the request header. The value directs the receiver of the request to return a response unconditionally, whether positive or negative, to that request. Contrast with *exception response* and *no response.*

definition statement. (1) In VTAM, the statement that describes an element of the network. (2) In NCP, a type of instruction that defines a resource to the NCP. See also *macro instruction.*

direct access storage device (DASD). A device in which the access time is effectively independent of the location of the data. For example, a disk.

directory. In VM, a control program (CP) disk that defines each virtual machine's normal configuration.

disabled. In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is temporarily not ready to establish LU-LU sessions. An initiate request for a session with a disabled logical unit (LU) can specify that the session be queued by the SSCP until the LU becomes enabled. The LU can separately indicate whether this applies to its ability to act as a primary logical unit (PLU) or a secondary logical unit (SLU). See also *enabled* and *inhibited.*

domain operator. In a multiple-domain network, the person or program that controls the operation of the resources controlled by one system services control point. Contrast with *network operator* (2).

DR. (1) In NCP and CCP, dynamic reconfiguration. (2) In SNA, definite response.

dump. (1) Computer printout of storage. (2) To write the contents of all or part of storage to an external medium as a safeguard against errors or in connection with debugging. (3) (ISO) Data that have been dumped.

dynamic reconfiguration (DR). The process of changing the network configuration (peripheral PUs and LUs) without regenerating complete configuration tables.

ECB. Event control block.

ECL. Electronic cabling link.

emulation mode. The function of a network control program that enables it to perform activities equivalent to those performed by a transmission control unit. Contrast with *network control mode.*

Emulation Program (EP). An IBM control program that allows a channel-attached 3705 or 3725 communication controller to emulate the functions of an IBM 2701 Data Adapter Unit, an IBM 2702

Transmission Control, or an IBM 2703 Transmission Control. See also *network control program.*

enabled. In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is now ready to establish LU-LU sessions. The LU can separately indicate whether this prevents it from acting as a primary logical unit (PLU) or as a secondary logical unit (SLU). See also *disabled* and *inhibited.*

EP. Emulation Program.

ER. (1) Explicit route. (2) Exception response.

event control block (ECB). A control block used to represent the status of an event.

exception response (ER). In SNA, a negative response shown as a value in the form-of-response-requested field of a request header (RH). An exception response is sent only if a request is unacceptable as received or cannot be processed. Contrast with *definite response* and *no response.* See also *negative response.*

EXEC. In a VM operating system, a user-written command file that contains CMS commands, other user-written commands, and execution control statements, such as branches.

explicit route (ER). In SNA, the path control network elements, including a specific set of one or more transmission groups, that connect two subarea nodes. An explicit route is identified by an origin subarea address, a destination subarea address, an explicit route number, and a reverse explicit route number. Contrast with *virtual route (VR).* See also *path* and *route extension.*

FASTRUN. One of several options available with the NCP/EP Definition Facility (NDF) that indicates only the syntax is to be checked in generation definition statements.

frame. (1) The unit of transmission in some local area networks, including the IBM Token-Ring Network. It includes delimiters, control characters, information, and checking characters. (2) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures.

generalized path information unit trace (GPT). A record of the flow of path information units (PIUs) exchanged between the network control program and its attached resources. PIU trace records consist of up to 44 bytes of transmission header (TH), request/response header (RH), and request/response unit (RU) data.

generation. The process of assembling and link editing definition statements so that resources can be identified to all the necessary programs in a network.

generation definition. The definition statement of a resource used in generating a program.

GPT. Generalized path information unit trace.

half-session. In SNA, a component that provides FMD services, data flow control, and transmission control for one of the sessions of a network addressable unit (NAU). See also *primary half-session* and *secondary half-session.*

host processor. (1) (TC97) A processor that controls all or part of a user application network. (2) In a network, the processing unit in which the data communication access method resides. (3) In an SNA network, the processing unit that contains a system services control point (SSCP).

inactive. In VTAM, describes the state of a resource that has not been activated or for which the VARY INACT command has been issued. Contrast with *active.* See also *inoperative.*

inhibited. In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is not ready to establish LU-LU sessions. An initiate request for a session with an inhibited LU will be rejected by the SSCP. The LU can separately indicate whether this applies to its ability to act as a primary logical unit (PLU) or as a secondary logical unit (SLU). See also *enabled* and *disabled.*

inoperative. The condition of a resource that has been active, but is not. The resource may have failed, received an INOP request, or is suspended while a reactivate command is being processed. See also *inactive.*

interface. * A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs.

intermediate routing node (IRN). In SNA, a subarea node with intermediate routing function. A subarea node may be a boundary node, an intermediate routing node, both, or neither, depending on how it is used in the network.

IRN. Intermediate routing node.

JCL. Job control language.

job control language (JCL). * A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

keyword. (1) * One of the predefined words of an artificial language. (2) One of the significant and informative words in a title or document that describes the content of that document. (3) A symbol that identifies a parameter. (4) A part of a command operand that consists of a specific character string (such as DSNAME=).

line. See *communication line.*

link. In SNA, the combination of the link connection and the link stations joining network nodes; for example: (1) a System/370 channel and its associated protocols, (2) a serial-by-bit connection under the control of Synchronous Data Link Control (SDLC). A link connection is the physical medium of transmission. A link, however, is both logical and physical. Synonymous with *data link.*

link-attached. In VTAM, pertaining to devices that are physically connected by a telecommunication line. Synonymous with *remote.* Contrast with *channel-attached.*

load module. (ISO) A program unit that is suitable for loading into main storage for execution; it is usually the output of a linkage editor.

local address. In SNA, an address used in a peripheral node in place of an SNA network address and transformed to or from an SNA network address by the boundary function in a subarea node.

local-attached. Deprecated term for *channel-attached.*

logical unit (LU). In SNA, a port through which an end user accesses the SNA network in order to communicate with another end user and through which the end user accesses the functions provided by system services control points (SSCPs). An LU can support at least two sessions—one with an SSCP and one with another LU—and may be capable of supporting many sessions with other logical units. See also *network addressable unit (NAU), peripheral LU, physical unit (PU), system services control point (SSCP), primary logical unit (PLU), and secondary logical unit (SLU).* Contrast with *physical unit (PU).*

logical unit (LU) services. In SNA, capabilities in a logical unit to: (1) receive requests from an end user and, in turn, issue requests to the system services control point (SSCP) in order to perform the requested functions, typically for session initiation; (2) receive requests from the SSCP, for example to activate LU-LU sessions via Bind Session requests; and (3) provide session presentation and other services for LU-LU sessions. See also *physical unit (PU) services.*

LU. Logical unit.

macro instruction. (1) * (ISO) An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language. The macro instruction may also specify values for parameters in the instructions that are to replace it. (2) In assembler programming, an assembler language statement that causes the assembler to process a predefined set of statements called a macro definition. The statements normally produced from the macro definition replace the macro instruction in the program. See also *definition statement.*

maintenance and operator subsystem (MOSS). A subsystem of the 3725 Communication Controller that contains a processor and operates independently of the rest of the controller. It loads and supervises the 3725, runs problem determination procedures, and assists in maintaining both hardware and software.

MDR. Miscellaneous data record.

message. In VTAM, the amount of FM data transferred to VTAM by the application program with one SEND request.

migration. Installing a new version or release of a program when an earlier version or release is already in place.

miscellaneous data record (MDR). A record of a network hardware error recorded by the NCP and sent to the VTAM host that owns the failing component. Then VTAM writes the error on the operating system error data set.

MOSS. Maintenance and operator subsystem.

Multiple Virtual Storage (MVS). An IBM program product whose full name is the Operating System/Virtual Storage (OS/VS) with Multiple Virtual Storage/System Product for System/370. It is a software operating system controlling the execution of programs.

MVS. Multiple Virtual Storage operating system.

NAU. Network addressable unit.

NCP. (1) Network Control Program (IBM program product). Its full name is Advanced Communications Function for the Network Control Program. (2) Network control program (general term).

NCP/EP definition facility (NDF). A program that is part of System Support Programs (SSP) and is used to generate a partitioned emulation programming (PEP) load module or a load module for a Network Control Program (NCP) or for an Emulation Program (EP).

NCP Subset. Advanced Communications Function for Network Control Program (NCP) V4 Subset. An IBM licensed program that is a subset of NCP. It operates only on IBM 3720 Communication Controllers with certain capacity limitations such as number of scanners, lines, and channel adapters supported.

NCP/Token-Ring interconnection (NTRI). An NCP function that allows a communication controller to attach to the IBM Token-Ring Network by providing a basic boundary network node interface.

NDF. NCP/EP definition facility.

negative response. In SNA, a response indicating that a request did not arrive successfully or was not processed successfully by the receiver. Contrast with *positive response*. See *exception response.*

network. (1) (TC97) An interconnected group of nodes. (2) In data processing, a user application network. See *path control network, public network, SNA network,* and *user application network.*

network address. In SNA, an address, consisting of subarea and element fields, that identifies a link, a link station, or a network addressable unit. Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See *local address.* See also *network name.*

network addressable unit (NAU). In SNA, a logical unit, a physical unit, or a system services control point. It is the origin or the destination of information transmitted by the path control network. Each NAU has a network address that represents it to the path control network. See also *network name, network address,* and *path control network.*

network control (NC). In SNA, an RU category used for requests and responses exchanged between physical units (PUs) for such purposes as activating and deactivating explicit and virtual routes and sending load modules to adjacent peripheral nodes. See also *data flow control layer* and *session control.*

network control mode. The functions of a network control program that enable it to direct a communication controller to perform activities such as polling, device addressing, dialing, and answering. Contrast with *emulation mode.*

Network Control Program (NCP). An IBM program product that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. Its full name is Advanced Communications Function for the Network Control Program.

network control program. A program, generated by the user from a library of IBM-supplied modules, that controls the operation of a communication controller.

networking. In a multiple-domain network, communication among domains.

network name. (1) In SNA, the symbolic identifier by which end users refer to a network addressable unit (NAU), a link, or a link station. See also *network address.* (2) In a multiple-domain network, the name of the APPL statement defining a VTAM application program is its network name and it must be unique across domains. Contrast with *ACB name.* See *uninterpreted name.*

network operator. (1) A person or program responsible for controlling the operation of all or part of a network. (2) The person or program that controls all the domains in a multiple-domain network. Contrast with *domain operator.*

network performance analyzer (NPA). An option of NCP that collects performance data about devices. The data is recorded by NPM.

Network Routing Facility (NRF). An IBM program product that resides in the NCP, which provides a path for messages between terminals, and routes messages over this path without going through the host processor.

NIB. Node initialization block.

node initialization block (NIB). In VTAM, a control block associated with a particular node or session that contains information used by the

application program to identify the node or session and to indicate how communication requests on a session are to be handled by VTAM.

no response (NR). In SNA, a value in the form-of-response-requested field of the request header (RH) indicating that no response is to be returned to the request, whether or not the request is received and processed successfully. Contrast with *definite response* and *exception response.*

NPSI. X.25 NCP Packet Switching Interface.

NTRI. NCP/Token-Ring interconnection.

operator. A person who operates a machine. See *network operator.*

PAB. Process anchor block.

page. (1) The portion of a panel that is shown on a display surface at one time. (2) To move back and forth among the pages of a multiple-page panel. See also *scroll.* (3) (ISO) In a virtual storage system, a fixed-length block that has a virtual address and that can be transferred between real storage and auxiliary storage. (4) To transfer instructions, data, or both between real storage and external page or auxiliary storage.

partitioned emulation programming (PEP) extension. A function of a network control program that enables a communication controller to operate some telecommunication lines in network control mode while simultaneously operating others in emulation mode.

path. (1) In SNA, the series of path control network components (path control and data link control) that are traversed by the information exchanged between two network addressable units (NAUs). A path consists of a virtual route and its route extension, if any. See also *explicit route.* (2) In defining a switched major node, a potential dial-out port that can be used to reach a physical unit.

path control (PC) layer. In SNA, the layer that manages the sharing of link resources of the SNA network and routes basic information units (BIUs) through it. Path control routes message units between network addressable units (NAUs) in the network and provides the paths between them. It converts the BIUs from transmission control (possibly segmenting them) into path information units (PIUs) and exchanges basic transmission units (BTUs) and one or more PIUs with data link control. See also *BIU segment, blocking of PIUs, data link control layer,* and *transmission control layer.*

path control (PC) network. In SNA, the part of the SNA network that includes the data link control and path control layers. See *SNA network* and *user application network.* See also *boundary function.*

path information unit (PIU). In SNA, a message unit consisting of a transmission header (TH) alone, or of a TH followed by a basic information unit (BIU) or a BIU segment. See also *transmission header.*

PC. Path control.

PEP. Partitioned emulation programming.

peripheral LU. In SNA, a logical unit representing a peripheral node.

peripheral PU. In SNA, a physical unit representing a peripheral node.

physical connection. In VTAM, a point-to-point connection or multipoint connection.

physical unit (PU). In SNA, one of three types of network addressable units (NAUs). Each node of an SNA network contains a physical unit (PU) that manages and monitors the resources (such as attached links) of a node, as requested by a system services control point (SSCP) via an SSCP-PU session. An SSCP activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links. See also *peripheral PU, physical unit (PU) type,* and *subarea PU.*

physical unit (PU) services. In SNA, the components within a physical unit (PU) that provide configuration services and maintenance services for SSCP-PU sessions. See also *logical unit (LU) services.*

physical unit (PU) type. In SNA, the classification of a physical unit (PU) according to the type of node in which it resides. The PU type is the same as its node type; that is, a type 1 PU resides in a type 1 node, and so forth.

PIU. Path information unit.

PLU. Primary logical unit.

positive response. A response indicating that a request was received and processed. Contrast with *negative response.*

primary half-session. In SNA, the half-session that sends the session activation request. See also

primary logical unit. Contrast with *secondary half-session.*

primary logical unit (PLU). In SNA, the logical unit (LU) that contains the primary half-session for a particular LU-LU session. Each session must have a PLU and secondary logical unit (SLU). The PLU is the unit responsible for the bind and is the controlling LU for the session. A particular LU may contain both primary and secondary half-sessions for different active LU-LU sessions. Contrast with *secondary logical unit (SLU).*

problem determination. The process of identifying the source of a problem; for example, a program component, a machine failure, telecommunication facilities, user or contractor-installed programs or equipment, an environment failure such as a power loss, or a user error.

process anchor block (PAB). In VTAM, a process scheduling services dispatch point.

PU. Physical unit.

public network. A network established and operated by communication common carriers or telecommunication Administrations for the specific purpose of providing circuit-switched, packet-switched, and leased-circuit services to the public. Contrast with *user-application network.*

Recommendation X.21 (Geneva 1980). A Consultative Committee on International Telegraph and Telephone (CCITT) recommendation for a general purpose interface between data terminal equipment and data circuit equipment for synchronous operations on a public data network.

Recommendation X.25 (Geneva 1980). A Consultative Committee on International Telegraph and Telephone (CCITT) recommendation for the interface between data terminal equipment and packet-switched data networks. See also *packet switching.*

release. For VTAM to relinquish control of resources (communication controllers or physical units). See also *resource takeover.* Contrast with *acquire (2).*

remote. Synonym for *link-attached.*

remote modem self-test (RST). A check on hardware to identify a field-replaceable unit that is failing.

request header (RH). In SNA, control information preceding a request unit (RU). See also *request/response header (RH).*

request unit (RU). In SNA, a message unit that contains control information such as a request code or FM headers, end-user data, or both.

request/response header (RH). In SNA, control information, preceding a request/response unit (RU), that specifies the type of RU (request unit or response unit) and contains control information associated with that RU.

request/response unit (RU). In SNA, a generic term for a request unit or a response unit. See also *request unit (RU)* and *response unit.*

resource. (1) Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In NetView, any hardware or software that provides function to the network.

resource takeover. In VTAM, action initiated by a network operator to transfer control of resources from one domain to another. See also *acquire (2)* and *release.* See *takeover.*

response header (RH). In SNA, a header, optionally followed by a response unit (RU), that indicates whether the response is positive or negative and that may contain a pacing response. See also *negative response, pacing response,* and *positive response.*

response unit (RU). In SNA, a message unit that acknowledges a request unit; it may contain prefix information received in a request unit. If positive, the response unit may contain additional information (such as session parameters in response to Bind Session), or if negative, contains sense data defining the exception condition.

return code. * A code [returned from a program] used to influence the execution of succeeding instructions.

RH. Request/response header.

ring. A network configuration where a series of attaching devices are connected by unidirectional transmission links to form a closed path.

route extension (REX). In SNA, the path control network components, including a peripheral link, that make up the portion of a path between a subarea node and a network addressable unit (NAU)

in an adjacent peripheral node. See also *path, explicit route (ER), virtual route (VR).*

RST. Remote modem self-test.

RU. Request/response unit.

RU chain. In SNA, a set of related request/response units (RUs) that are consecutively transmitted on a particular normal or expedited data flow. The request RU chain is the unit of recovery: if one of the RUs in the chain cannot be processed, the entire chain is discarded. Each RU belongs to only one chain, which has a beginning and an end indicated via control bits in request/response headers within the RU chain. Each RU can be designated as first-in-chain (FIC), last-in-chain (LIC), middle-in-chain (MIC), or only-in-chain (OIC). Response units and expedited-flow request units are always sent as only-in-chain.

scanner interface trace (SIT). A record of the activity within the communication scanner processor (CSP) for a specified data link between a 3725 Communication Controller and a resource.

scroll. To move all or part of the display image vertically to display data that cannot be observed within a single display image. See also *page (2).*

secondary half-session. In SNA, the half-session that receives the session-activation request. See also *secondary logical unit (SLU).* Contrast with *primary half-session.*

secondary logical unit (SLU). In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. An LU may contain secondary and primary half-sessions for different active LU-LU sessions. Contrast with *primary logical unit (PLU).*

secondary logical unit (SLU) key. A key-encrypting key used to protect a session cryptography key during its transmission to the secondary half-session.

session control (SC). In SNA, (1) One of the components of transmission control. Session control is used to purge data flowing in a session after an unrecoverable error occurs, to resynchronize the data flow after such an error, and to perform cryptographic verification. (2) A request unit (RU) category used for requests and responses exchanged between the session control components of a session and for session activation and deactivation requests and responses.

SIT. Scanner interface trace.

SLU. Secondary logical unit.

SNA. Systems Network Architecture.

SNA network. The part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network addressable units (NAUs), boundary function components, and the path control network.

SSCP. System services control point.

SSP. System Support Programs (IBM program product). Its full name is Advanced Communications Function for System Support Programs.

subarea PU. In SNA, a physical unit (PU) in a subarea node.

subsystem. A secondary or subordinate system, usually capable of operating independent of, or asynchronously with, a controlling system.

system services control point (SSCP). In SNA, a focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for end users of the network. Multiple SSCPs, cooperating as peers, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its domain.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

System Support Programs (SSP). An IBM program product, made up of a collection of utilities and small programs, that supports the operation of the NCP.

takeover. The process by which the failing active subsystem is released from its Extended Recovery Facility (XRF) sessions with terminal users and replaced by an alternate subsystem. See *resource takeover.*

task. A basic unit of work to be accomplished by a computer. The task is usually specified to a control program in a multiprogramming or multiprocessing environment.

TCAM. (1) Telecommunications Access Method. (2) The IBM program product whose full name is Advanced Communications Function for TCAM and that provides queued message handling. TCAM Versions 1 and 2 are access methods, but TCAM Version 3 is a message handling subsystem.

telecommunication line. Any physical medium such as a wire or microwave beam, that is used to transmit data. Synonymous with *transmission line.*

terminal. A device that is capable of sending and receiving information over a link; it is usually equipped with a keyboard and some kind of display, such as a screen or a printer.

TH. Transmission header.

token. A sequence of bits passed from one device to another along the network. When the token has data appended to it, it becomes a frame.

token ring. A network, having a ring topology, that passes tokens from one attaching device to another. For example, the IBM Token-Ring Network.

transmission control (TC) layer. In SNA, the layer within a half-session that synchronizes and paces session-level data traffic, checks session sequence numbers of requests, and enciphers and deciphers end-user data. Transmission control has two components: the connection point manager and session control. See also *half-session.*

transmission header (TH). In SNA, control information, optionally followed by a basic information unit (BIU) or a BIU segment, that is created and used by path control to route message units and to control their flow within the network. See also *path information unit.*

transmission line. Synonym for *telecommunication line.*

uninterpreted name. In SNA, a character string that a system services control point (SSCP) is able to convert into the network name of a logical unit (LU). Typically, an uninterpreted name is used in a logon or Initiate request from a secondary logical unit (SLU) to identify the primary logical unit (PLU) with which the session is requested.

user. Anyone who requires the services of a computing system.

user-application network. A configuration of data processing products, such as processors, controllers, and terminals, established and operated by users for the purpose of data processing or information exchange, which may use services offered by communication common carriers or telecommunication Administrations. Contrast with *public network.*

user-written generation application. A user-written program that runs with the NCP/EP definition facility (NDF) during NCP generation. It processes definition statements and operands.

variable. In NetView, a character string beginning with & that is coded in a command list and is assigned a value during execution of the command list.

virtual machine. A functional simulation of a computer and its associated devices.

Virtual Machine (VM). A program product whose full name is the Virtual Machine/System Product (VM/SP). It is a software operating system that manages the resources of a real processor to provide virtual machines to end users. As a time-sharing system control program, it consists of the virtual machine control program (CP), the conversational monitor system (CMS), the group control system (GCS), and the interactive problem control system (IPCS).

virtual route (VR). In SNA, a logical connection (1) between two subarea nodes that is physically realized as a particular explicit route, or (2) that is contained wholly within a subarea node for intra-node sessions. A virtual route between distinct subarea nodes imposes a transmission priority on the underlying explicit route, provides flow control through virtual-route pacing, and provides data integrity through sequence numbering of path information units (PIUs). See also *explicit route (ER), path,* and *route extension.*

virtual route (VR) pacing. In SNA, a flow control technique used by the virtual route control component of path control at each end of a virtual route to control the rate at which path information units (PIUs) flow over the virtual route. VR pacing can be adjusted according to traffic congestion in any of the nodes along the route. See also *pacing* and *session-level pacing.*

virtual storage. (ISO) The notion of storage space that may be regarded as addressable main storage

by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, not by the actual number of main storage locations.

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

Virtual Storage Extended (VSE). An IBM program product whose full name is the Virtual Storage Extended/Advanced Function. It is a software operating system controlling the execution of programs.

Virtual Telecommunications Access Method (VTAM). An IBM program product that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VIT. VTAM internal trace.

VM. Virtual Machine operating system. Its full name is Virtual Machine/System Product.

VM/SP. Virtual Machine/System Product operating system. Synonym for *VM*.

VR. Virtual route.

VSAM. Virtual Storage Access Method.

VSE. Virtual Storage Extended operating system.

VTAM. Virtual Telecommunications Access Method (IBM program product). Its full name is Advanced Communications Function for the Virtual Telecommunications Access Method.

VTAM internal trace (VIT). A trace used in VTAM to collect data on channel I/O, use of locks, and storage management services.

VTAM operator command. A command used to monitor or control a VTAM domain.

XID. A data link control command and response passed between adjacent nodes that allows the two nodes to exchange identification and other information necessary for operation over the data link.

X.21. See *Recommendation X.21 (Geneva 1980).*

X.25. See *Recommendation X.25 (Geneva 1980).*

X.25 NCP Packet Switching Interface (NPSI). The X.25 Network Control Program Packet Switching Interface, which is an IBM program product that allows SNA users to communicate over packet-switched data networks that have interfaces complying with Recommendation X.25 (Geneva 1980) of the International Telegraph and Telephone Consultative Committee (CCITT). It allows SNA programs to communicate with SNA equipment or with non-SNA equipment over such networks. In addition, this product may be used to attach native X.25 equipment to SNA host systems without a packet network. See also *Recommendation X.25 (Geneva 1980).*

Index

A

access method loader facility
  MVS 3-1
  VM 6-1
  VSE 9-1
ALIGN2 option
  in EXEC, for VM 5-4, 5-10
  in JCL, for MVS 2-3, 2-6
ASMLIST data set, for MVS 1-8
ASMLIST file, for VM 4-8
ASMOBJ data set, for MVS 1-8
ASMOBJ file, for VM 4-8
ASMSRCE data set, for MVS 1-8
ASMSRCE file, for VM 4-8
ASSEMBLY parameter
  MVS 1-9
  VM 4-9
ASSIGN statement, for VSE 9-4

B

BUILD definition statement
  LENAME operand, for VSE 7-10
  NCPCA operand
    MVS 3-1
    VM 6-1
    VSE 9-1
  NEWNAME operand
    MVS 1-11
    VM 4-11
    VSE 7-8

C

ccname specification, for MVS 3-4
channel adapters
  MVS 3-1
  VM 6-1
  VSE 9-1
CMS file, loader for VM 6-3
communication controller
  identifying for loading
    MVS 3-4
    VM 6-4
    VSE 9-5
  initial test routine, 3705
    MVS 3-1, 3-2, 3-7, 3-9
    VM 6-1, 6-3, 6-6, 6-7
    VSE 9-1, 9-2, 9-7, 9-8
  loading requirements
    MVS 3-2
    VM 6-2
    VSE 9-2
completion messages, loading
  MVS 3-3
  VM 6-3
  VSE 9-3
CP DEFINE STORAGE command, for VM 4-10
cuu specification, for VM 6-4

D

DASD file, loader for VSE 9-2
DASD partitioned data set, loader for MVS 3-2
DASD, work space requirements
  MVS 1-5
  VM 4-5
  VSE 7-4
data sets for MVS, descriptions
  ASMLIST 1-8
  ASMOBJ 1-8
  ASMSRCE 1-8
  DBWORKFL 1-6
  GENDECK 1-6
  LNKSTMT 1-7
  NEWDEFN 1-7, 1-17
  OBJxxxx 1-8
  PRINTER 1-7
  STEPLIB 1-6
  SYSIN 3-4
  SYSLIB 1-7
  SYSLIN 1-7
  SYSLMOD 1-8
  SYSPRINT 1-7, 3-4
  SYSPUNCH 1-7
  SYSUT1 1-7, 3-4
  SYSUT3 3-4
  TBL1LIST 1-7
  TBL1OBJ 1-7
  TBL1SRCE 1-6
  TBL2LIST 1-7
  TBL2OBJ 1-7
  TBL2SRCE 1-6
  ULIB 1-8, 2-11
data sets, specifying for MVS
  See also data sets for MVS, descriptions
  for generation 1-6
  for loading 3-4
DBWORKFL data set, for MVS 1-6
DBWORKFL file, for VM 4-6
DBWRKFL file, for VSE 7-5
ddnames

See data sets for MVS, descriptions
DIAG operand
MVS 3-4
VM 6-4
VSE 9-5
DLBL statement, for VSE 9-4
dtfnames
See file names for VSE, descriptions
dynamic reconfiguration generation
description
MVS 1-16
VM 4-16
VSE 7-11
example of EXEC, for VM 5-19
example of JCL
MVS 2-12
VSE 8-7

E

error message summary
MVS 1-17
VM 4-17
VSE 7-12
error messages
for generation
MVS 1-17
VM 4-17
VSE 7-12
for loading
MVS 3-3
VM 6-3
VSE 9-3
EXECs, examples for VM
for generation 5-1
for loading 6-6

F

FASTRUN generation
description
MVS 1-11
VM 4-11
VSE 7-10
example of EXEC, for VM 5-2
example of JCL
MVS 2-2
VSE 8-2
FASTRUN operand and parameter
MVS 1-11, 2-2
VM 4-11, 5-2
VSE 7-10, 8-2
file names for VM, descriptions
ASMLIST 4-8
ASMOBJ 4-8
ASMSRCE 4-8
DBWORKFL 4-6
GENDECK 4-6
LNKSTMT 4-7
NEWDEFN 4-7, 4-17
OBJxxxx 4-8
PRINTER 4-7
STEPLIB 4-6
SYSIN 6-3
SYSLIB 4-7
SYSLIN 4-7
SYSLMOD 4-8
SYSPRINT 4-7, 6-3
SYSPUNCH 4-7
SYSUT1 4-7, 6-3
SYSUT3 6-3
TBL1LIST 4-7
TBL1OBJ 4-7
TBL1SRCE 4-6
TBL2LIST 4-7
TBL2OBJ 4-7
TBL2SRCE 4-6
ULIB 4-8, 5-18
file names for VSE, descriptions
DBWRKFL 7-5
IJSYSIN 7-5
IJSYSPH 7-5
files, specifying
for generation
VM 4-6
VSE 7-5
for loading
VM 6-3
VSE 9-4

G

GENDECK data set, for MVS 1-6
GENDECK file, for VM 4-6
GENEND definition statement
MVS 1-14, 2-11
VM 4-14, 5-18
VSE 7-10, 8-6
generation
controlling
MVS 1-6
VM 4-6
VSE 7-5
definition
MVS 1-6, 1-7
VM 4-6, 4-7
VSE 7-5
generation, EXECs for VM 5-1
job control language
MVS 2-1

VSE 8-1
listings and error messages
MVS 1-17
VM 4-17
VSE 7-12
listings, sample
MVS 1-18
VM 4-18
VSE 7-12
procedure, description
MVS 1-2
VM 4-2
VSE 7-2
report
MVS 1-18
VM 4-18
VSE 7-12
steps for
MVS 1-2
VM 4-2
VSE 7-2
types of
MVS 1-1, 1-6
VM 4-1, 4-6
VSE 7-1, 7-5
validation listing
MVS 1-9
VM 4-9
VSE 7-6
GETVIS region, for VSE
for storage manager data 7-4
GLOBAL LOADLIB statement, for VM 6-3

H

hardware and software combinations vi
host processor, requirements
for generation
MVS 1-1
VM 4-1
VSE 7-1
for loading
MVS 3-2
VM 6-2
VSE 9-2

I

IBM 3705 Communications Controller
identifying for loading
MVS 3-4
VM 6-4
VSE 9-5
initial test routine, loading
MVS 3-1, 3-2, 3-4, 3-7, 3-9
VM 6-1, 6-3, 6-4, 6-6, 6-7
VSE 9-1, 9-2, 9-5, 9-7, 9-8
loading requirements
MVS 3-2
VM 6-2
VSE 9-2
IBM 3720 Communication Controller
identifying for loading
MVS 3-4
VM 6-4
VSE 9-5
loading requirements
MVS 3-2
VM 6-2
VSE 9-2
IBM 3725 Communication Controller
identifying for loading
MVS 3-4
VM 6-4
VSE 9-5
loading requirements
MVS 3-2
VM 6-2
VSE 9-2
IJSYSIN file, for VSE 7-5
IJSYSPH file, for VSE 7-5
initial test routine
description
MVS 3-1
VM 6-1
VSE 9-1
DIAG statement
MVS 3-4
VM 6-4
VSE 9-5
example of control statements
MVS 3-7, 3-9
VM 6-6, 6-7
VSE 9-7, 9-8
input to loader
MVS 3-2
VM 6-3
VSE 9-2
introduction to manual
how to use vi
hardware and software combinations vi
organization vii
abbreviations ix
other manuals x
purpose v

J

job control language, examples
  for generation
    MVS 2-1
    VSE 8-1
  for loading
    MVS 3-7
    VSE 9-3, 9-6
job control statements, loading
  MVS 3-4, 3-7
  VSE 9-4, 9-6

L

LENAME operand, for VSE 7-10
LIBDEF statement, for VSE 9-4
LIBRARIAN, for VSE
  punching phases onto disk, loading 9-3
  step for generation 7-2
line count, defining
  MVS 1-9
  VM 4-9
  VSE 7-6
LINECNT parameter
  MVS 1-9
  VM 4-9
  VSE 7-6
link-edit
  See linkage editor step for generation
link-editing object code into phases, for VSE 9-8
linkage editor step for generation
  MVS 1-3
  VM 4-3
  VSE 7-2
listings from generation
  sample
    MVS 1-18
    VM 4-18
    VSE 7-12
LNKSTMT data set, for MVS 1-7
LNKSTMT file, for VM 4-7
load modules (NCP), naming
  MVS 1-11
  VM 4-11
load modules for loader utility
  MVS 3-2
  VM 6-2
LOAD statement
  MVS 3-2, 3-4
  VM 6-3, 6-4
  VSE 9-2, 9-5
loader utility
  description
    MVS 3-1, 3-2
    VM 6-1, 6-2
    VSE 9-1, 9-2
  input to
    MVS 3-2
    VM 6-3
    VSE 9-2
  load modules
    MVS 3-2
    VM 6-2
  output from
    MVS 3-3
    VM 6-3
    VSE 9-3
  phases, for VSE 9-2
loading
  controlling
    MVS 3-4
    VM 6-3
    VSE 9-4
  EXECs, for VM 6-6
  job control language
    MVS 3-4, 3-7
    VSE 9-3, 9-4, 9-6
  loader utility
    MVS 3-1, 3-2
    VM 6-1, 6-2
    VSE 9-1, 9-2
LOADMOD operand
  MVS 3-4
  VM 6-4
  VSE 9-5

M

message data set, loader for MVS 3-3
message logical unit, loader for VSE 9-3

N

naming NCP load modules
  MVS 1-11
  VM 4-11
naming NCP phases, for VSE 7-8
naming resources
  MVS 1-9
  VM 4-9
  VSE 7-6
NCP
  See network control program, compatibilities
NCP load module
  input data set for loading, for MVS 3-2
  input file for loading, for VM 6-3
  naming
    MVS 1-11

VM 4-11
NCP phases, for VSE
input file for loading 9-2
naming 7-8
NCP subset
See network control program subset, compatibilities
NCP/EP definition facility
introduction
MVS 1-1
VM 4-1
VSE 7-1
performance considerations
MVS 1-5
VM 4-5
VSE 7-4
NCP/PEP generation
description
MVS 1-12
VM 4-12
VSE 7-10
example of EXEC, for VM 5-4, 5-10
example of JCL
MVS 2-3, 2-6
VSE 8-3
NCP/Token-Ring interconnection
MVS 1-12, 2-3, 2-6
VM 4-12, 5-4, 5-10
NCPCA operand
MVS 3-1
VM 6-1
VSE 9-1
NDF
See NCP/EP definition facility
NDF standard attachment facility
introduction
MVS 1-1
VM 4-1
NEWDEFN data set, for MVS 1-7, 1-17
NEWDEFN file, for VM 4-7, 4-17
NEWDEFN operand
MVS 1-14, 1-17
VM 4-14, 4-17
steps in generation
MVS 1-2
VM 4-2
user-written code generation, description
MVS 1-13
VM 4-13
user-written code generation, example
MVS 2-9
VM 5-17
user-written generation applications
MVS 1-13
VM 4-13
USERGEN operand
MVS 1-14
VM 4-14
NDF SYSLIB chain
See SYSLIB chain
network control program subset, compatibilities
with controller vi
with EP for PEP vi
with SSP vi
network control program, compatibilities
with controller vi
with EP for PEP vi
with SSP vi
NEWDEFN data set, for MVS 1-7, 1-17
NEWDEFN file, for VM 4-7, 4-17
NEWDEFN operand
for NTRI
MVS 1-12, 2-3, 2-6
VM 4-12, 5-4, 5-10
for user-written code
MVS 1-14, 1-17, 2-9
VM 4-14, 4-17, 5-17
NEWNAME operand
MVS 1-11
VM 4-11
VSE 7-8
NTRI
See NCP/Token-Ring interconnection

O

object code, link editing for VSE 9-8
OBJxxxx data set, for MVS 1-8
OBJxxxx file, for VM 4-8
OPTIONS definition statement
FASTRUN operand
MVS 1-11, 2-2
VM 4-11, 5-2
VSE 7-10, 8-2
NEWDEFN operand
MVS 1-12, 1-14, 1-17, 2-3, 2-6, 2-9
VM 4-12, 4-14, 4-17, 5-4, 5-10, 5-17
USERGEN operand
MVS 1-14, 2-9
VM 4-14, 5-17
output listing, loader for VM 6-3

P

partitioned data set for loader, for MVS 3-2
performance considerations, generation
MVS 1-5
VM 4-5
VSE 7-4
phase names, for VSE 7-8
phases (NCP), naming for VSE 7-8
phases for loader utility, for VSE 9-2
phases, link editing object code into for VSE 9-8
PRINTER data set, for MVS 1-7

PRINTER file, for VM 4-7

R

REGION parameter, for MVS 1-10
region size for loading
  See virtual storage for loader
region size, defining
  See virtual storage, defining
resource name and network cross reference
  MVS 1-17
  VM 4-17
  VSE 7-12
resources, naming
  MVS 1-9
  VM 4-9
  VSE 7-6
return code
  for succeeding generation steps
    MVS 1-11
    VM 4-11
    VSE 7-9
  in generation report
    MVS 1-17
    VM 4-17
    VSE 7-12
  listing of
    MVS 1-11
    VM 4-11
return code summary
  PRINTER data set, for MVS 1-7
  PRINTER file, for VM 4-7
  step in generation
    MVS 1-3
    VM 4-3
REXX EXECs, for VM
  for generation 5-1
  for loading 6-6

S

SIZE parameter, for VSE 7-7
SRCHI code
  MVS 2-11
  VM 5-18
  VSE 8-6
SRCLO code
  MVS 2-11
  VM 5-18
  VSE 8-6
SSP
  See system support programs, compatibilities
SSP loader utility
  See loader utility
standard attachment facility
  See NDF standard attachment facility
STEPLIB data set, for MVS 1-6
STEPLIB file, for VM 4-6
storage for loader
  MVS 3-2
  VM 6-2
  VSE 9-2
storage manager
  MVS 1-5
  VM 4-5
  VSE 7-4
storage manager work data set, for MVS
  for standard attachment facility 1-5
  specifying 1-5, 1-6
storage manager work file
  for standard attachment facility, for VM 4-5
  specifying
    VM 4-5, 4-6
    VSE 7-4, 7-5, 7-7
storage, defining virtual
  MVS 1-10
  VM 4-10
  VSE 7-7
SYSIN data set, for MVS 3-4
SYSIN file, for VM 6-3
SYSLIB chain
  MVS 2-11
  VM 5-18
SYSLIB data set, for MVS 1-7
SYSLIB file, for VM 4-7
SYSLIN data set, for MVS 1-7
SYSLIN file, for VM 4-7
SYSLMOD data set, for MVS 1-8
SYSLMOD file, for VM 4-8
SYSLST logical unit, for VSE 9-3
SYSPRINT data set, for MVS
  for generating 1-7
  for loading 3-3
SYSPRINT file, for VM
  for generating 4-7, 6-3
  for loading 6-3
SYSPUNCH data set, for MVS 1-7
SYSPUNCH file, for VM 4-7
system support programs, compatibilities
  with controller vi
  with EP for PEP vi
  with NCP vi
SYSUT1 data set, for MVS 1-7
SYSUT1 file, for VM 4-7
SYSUT3 data set, for MVS 3-4
SYSUT3 file, for VM 6-3
SYSxxx specification, for VSE 9-5

T

table assemblies
  input data sets, for MVS 1-6
  input files
    VM 4-6
    VSE 7-5
  listing data sets, for MVS 1-7
  listing files, for VM 4-7
  output data sets, for MVS 1-7
  output files
    VM 4-7
    VSE 7-5
  steps in generation
    MVS 1-3
    VM 4-3
    VSE 7-2
  table 1 listing, block size
    MVS 1-5
    VM 4-5
TBL1LIST data set, for MVS 1-7
TBL1LIST file, for VM 4-7
TBL1OBJ data set, for MVS 1-7
TBL1OBJ file, for VM 4-7
TBL1SRCE data set, for MVS 1-6
TBL1SRCE file, for VM 4-6
TBL2LIST data set, for MVS 1-7
TBL2LIST file, for VM 4-7
TBL2OBJ data set, for MVS 1-7
TBL2OBJ file, for VM 4-7
TBL2SRCE data set, for MVS 1-6
TBL2SRCE file, for VM 4-6
3705 Communications Controller
  identifying for loading
    MVS 3-4
    VM 6-4
    VSE 9-5
  initial test routine, loading
    MVS 3-1, 3-2, 3-4, 3-7, 3-9
    VM 6-1, 6-3, 6-4, 6-6, 6-7
    VSE 9-1, 9-2, 9-5, 9-7, 9-8
  loading requirements
    MVS 3-2
    VM 6-2
    VSE 9-2
3725 Communication Controller
  identifying for loading
    MVS 3-4
    VM 6-4
    VSE 9-5
  loading requirements
    MVS 3-2
    VM 6-2
    VSE 9-2
3720 Communication Controller
  identifying for loading
    MVS 3-4
    VM 6-4
    VSE 9-5
  loading requirements
    MVS 3-2
    VM 6-2
    VSE 9-2

U

ULIB data set, for MVS 1-8, 2-11
ULIB file, for VM 4-8, 5-18
UNIT operand
  MVS 3-4
  VM 6-4
  VSE 9-5
user-written code generation
  description, using CSECTS
    MVS 1-12, 1-14
    VM 4-12, 4-14
    VSE 7-10
  description, using NDF standard attachment facility
    MVS 1-12, 1-13
    VM 4-12, 4-13
  example of EXEC for VM, using CSECTS 5-18
  example of EXEC for VM, using NDF standard attachment facility 5-17
  example of JCL for MVS, using NDF standard attachment facility 2-9
  example of JCL, using CSECTS
    MVS 2-11
    VSE 8-6
user-written generation applications
  MVS 1-13, 2-9
  VM 4-13, 5-17
user-written generation load modules
  MVS 1-13
  VM 4-13
USERGEN operand
  MVS 1-14, 2-9
  VM 4-14, 5-17

V

Virtual Storage Access Method (VSAM), for VSE
  defining cluster for work file 7-4
virtual storage for loader
  MVS 3-2
  VM 6-2
  VSE 9-2
virtual storage, defining
  MVS 1-10
  VM 4-10
  VSE 7-7
VM/SP commands, loading for VM 6-3, 6-6

VSAM
See Virtual Storage Access Method (VSAM), for VSE

W

work space requirements, DASD
MVS 1-5
VM 4-5
VSE 7-4

Numerics

3705 Communications Controller
identifying for loading
MVS 3-4
VM 6-4
VSE 9-5
initial test routine, loading
MVS 3-1, 3-2, 3-4, 3-7, 3-9
VM 6-1, 6-3, 6-4, 6-6, 6-7
VSE 9-1, 9-2, 9-5, 9-7, 9-8
loading requirements
MVS 3-2
VM 6-2
VSE 9-2
3720 Communication Controller
identifying for loading
MVS 3-4
VM 6-4
VSE 9-5
loading requirements
MVS 3-2
VM 6-2
VSE 9-2
3725 Communication Controller
identifying for loading
MVS 3-4
VM 6-4
VSE 9-5
loading requirements
MVS 3-2
VM 6-2
VSE 9-2

Network Control Program and System Support Programs Generation and Loading Guide

Publication No. SC30-3348-0

READER'S COMMENT FORM

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

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

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

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

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Dept. E03
P.O. Box 12195
Research Triangle Park, N.C. 27709-2195

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

SC30-3348-0

Printed in USA

SC30-3348-00