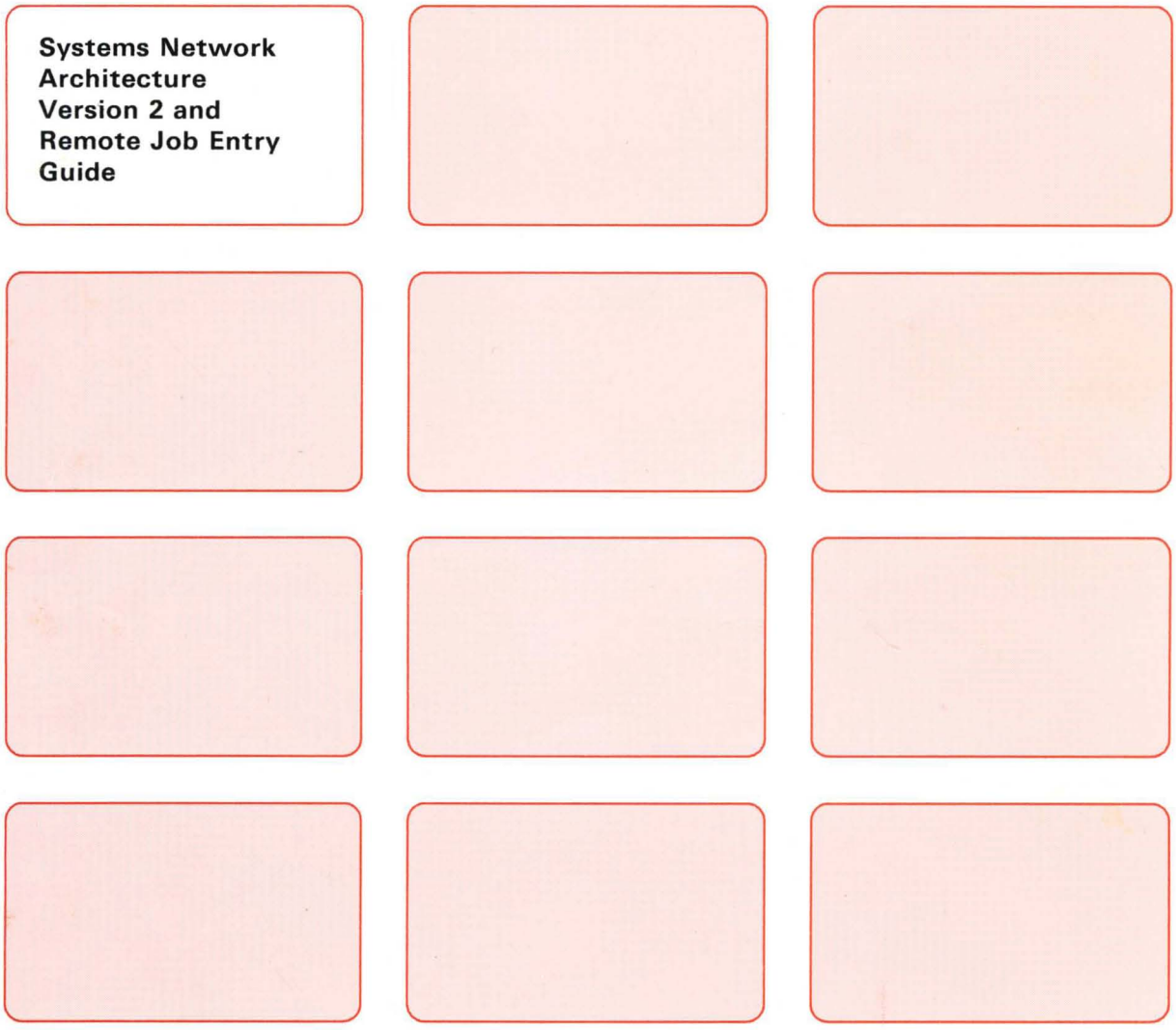
IBM

Series/1

SC34-0773-1

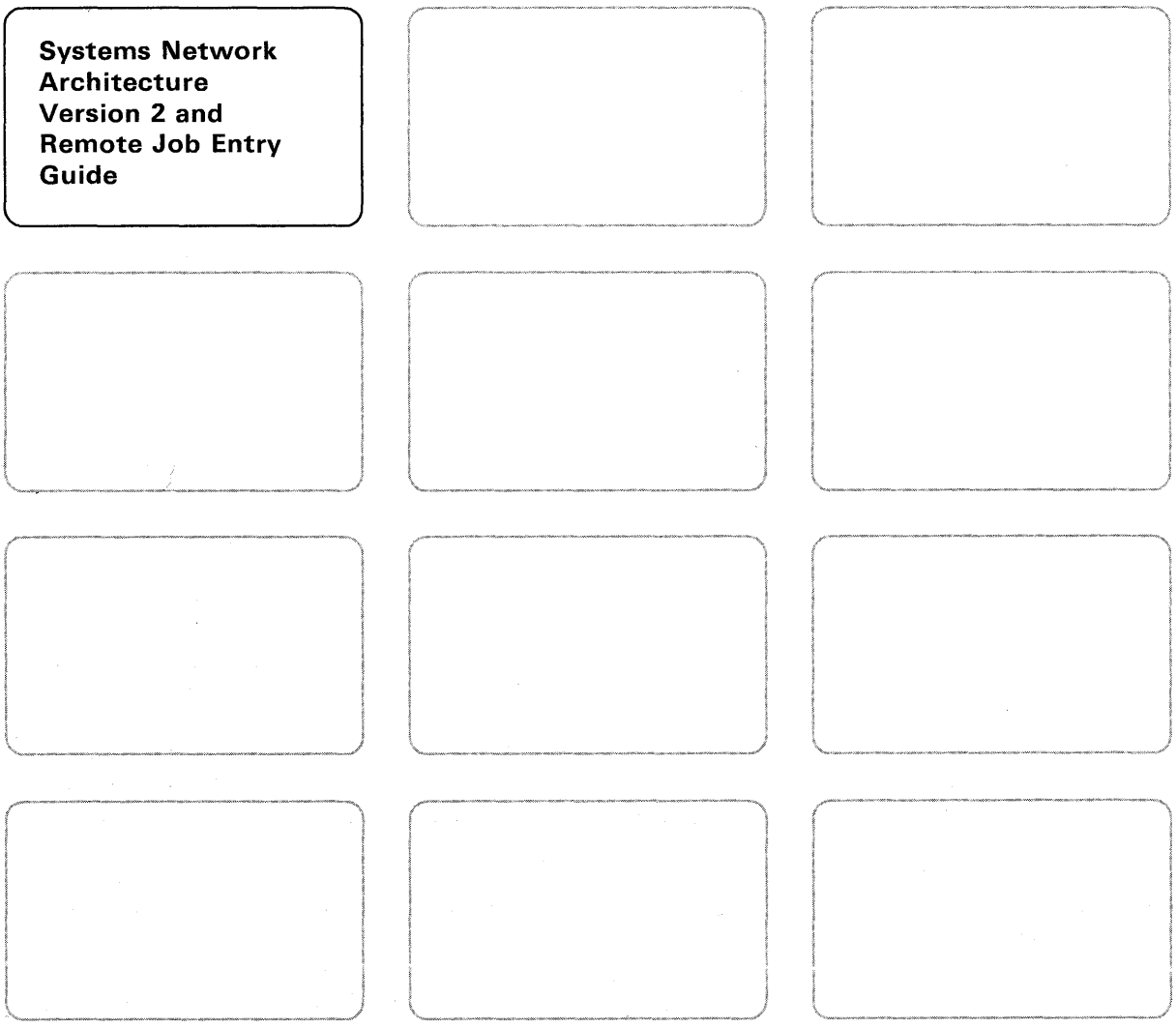
Event Driven Executive Systems Network Architecture Version 2 and Remote Job Entry Guide



IBM Series/1

SC34-0773-1

Event Driven Executive Systems Network Architecture Version 2 and Remote Job Entry Guide



Second Edition (October 1987)

This edition is a major revision of, and makes obsolete, SC34-0773-0 and technical newsletter SN34-0946. This edition applies to Version 2.1 of the IBM Series/1 Event Driven Executive Systems Network Architecture, Program Number 5719-XX9, and Remote Job Entry, Program Number 5719-SX2.

Use this publication only for the purpose stated in the section entitled "About This Book."

Changes are made periodically to the information herein; any such changes will be reported in subsequent revisions or Technical Newsletters. Changes are marked by a vertical bar in the left margin.

This material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

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

This publication could contain technical inaccuracies or typographical errors. A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to IBM Corporation, Information Development, Department 28B, (5414), P. O. Box 1328, Boca Raton, Florida 33429-1328. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Summary of Changes For Version 2.1

This document contains the following changes:

Shared SDLC Support

- Chapter 3, "Installing and Defining the SNA Network for Use with Shared SDLC" on page 3-1, a new chapter, contains information on installing and defining SNA for use with shared SDLC.
- Chapter 4, "Network Operation" on page 4-1 contains information about operating the network with shared SDLC.
- Appendix A, "Messages" on page A-1 contains new messages related to the shared SDLC support

Feature #2080 and Feature #2090 support

- "Shared SDLC Support" on page 1-3 and "SDLC Support" on page 1-2 contain information about how the feature #2080 and #2090 attachments are supported.

4956 J and K Processor

- Chapter 4, "Network Operation" on page 4-1 contains information about the the 4956 J and K processor support.

A vertical line in the left margin indicates new or changed material.

Contents

About This Book

This guide describes the services and functions provided by the IBM Series/1 Event Driven Executive Systems Network Architecture, Program Number 5719-XX9, and the IBM Series/1 Event Driven Executive Systems Network Architecture Remote Job Entry, Program Number 5719-SX2.

Audience

This publication addresses the following topics:

- The *application programmer* who has a need for SNA[1] services as provided through a set of high-level instructions available with Series/1 SNA. We assume that the application programmer has a basic knowledge of SNA and associated line protocols. Also, the application programmer should have experience in programming realtime applications using Event Driven Language instructions.
- The *application programmer* and the *operator* who have need to use the Series/1 remote job entry to SNA host computer capabilities provided with these program products.
- The *system programmer* who has SNA experience and who is familiar with the Event Driven Executive System.

How this Book is Organized

This book has 18 chapters and 9 appendixes:

- Chapter 1, "Introduction," gives an overview of SNA.
- Chapter 2, "Installing and Defining the SNA Network for Use with SDLC," gives information on how to install and define the SNA network for use with the SDLC support packaged with the SNA product.
- Chapter 3, "Installing and Defining the SNA Network for Use with Shared SDLC," gives information on how to install and define the SNA network for use with the shared SDLC support packaged with the EDX Version 6 product.
- Chapter 4, "Network Operation," gives information on how to operate the SNA network with SDLC and shared SDLC support.
- Chapter 5, "Activating a Session - NETHOST and NETINIT," describes establishing an SNA session.
- Chapter 6, "Sending Messages - NETPUT," describes transmitting messages from a Series/1 application program to a host application program.
- Chapter 7, "Receiving Messages - NETGET," describes receiving messages from the host by the Series/1 application program.
- Chapter 8, "Controlling Message Exchange - NETCTL," describes sending status information to the host application.

[1] SNA and Series/1 SNA used throughout this document refer to Event Driven Executive Systems Network Architecture.

- Chapter 9, "Terminating a Session - NETTERM," describes how to terminate a session.
- Chapter 10, "Message Recovery and Resynchronization," shows how to use resynchronization.
- Chapter 11, "Extended Error Information," contains tables showing return codes and the conditions causing them.
- Chapter 12, "SNA Base Support," discusses SNA base support as a subset of the total SNA architecture.
- Chapter 13, "SNA Protocols," describes protocols for communicating within the SNA network.
- Chapter 14, "SNA Command and Data Flows," shows examples of typical data and command exchanges possible during a session.
- Chapter 15, "Series/1 SNA Instruction Processing," describes the base SNA commands and subsequent protocol processing.
- Chapter 16, "Overview of SNA Remote Job Entry," describes using the $RJESNA utility for communication.
- Chapter 17, "Installing the Remote Job Entry Utility," outlines installation steps required to use $RJESNA.
- Chapter 18, "X.21 Circuit Switched Network Support," introduces X.21 circuit switched network support and explains how to make the connection when using SNA.
- Appendix A, "Messages," explains the messages you may receive while using SNA or X.21 network support.
- Appendix B, "SNA Sample Application to IMS/VS," shows IMS/VS installation, requirements, and execution of the sample application.
- Appendix C, "VTAM Considerations," contains a list of VTAM programming considerations.
- Appendix D, "Host Subsystem Considerations," describes using EDX SNA as a 3790-type controller in an IMS/VS or CICS/VS environment.
- Appendix E, "Node Definition Statements," describes the host system NCP node generation macros and parameters.
- Appendix F, "Data Area Descriptions," contains descriptions of units of information, request/response header, transmission header, exchange station identification, and buffers.
- Appendix G, "Sense Codes," describes the sense codes used by SNA support.
- Appendix H, "Series/1 Network Activation Procedure and Checklist," describes the network activation procedures and a checklist of items to be considered.
- Appendix I, "SNA Buffer Management," contains information about managing buffers.

Coding Examples and Illustrations

Coding examples are fully executable portions of complete programs that you can enter as shown.

Coding illustrations are nonexecutable portions of incomplete programs that show the correct format of all required parameters for an instruction. We indicate missing code within coding illustrations (code provided by you) by a series of three vertical dots.

Terms

The term **System/370** used in this text is applicable also to 303x, 308x, 309x, and 4300 Series host systems processors, unless noted otherwise.

The term **SNA** used in this text applies to Series/1 Systems Network Architecture support.

The term **3705** used in this text applies to Host Communications controllers 3704, 3705, and 3725.

The term **APPC** used in this text applies to advanced program-to-program communication. APPC describes the protocols used by programs in separate processors to communicate with each other when executing a (single) distributed transaction. APPC provides a set of architecturally-defined primitives that serve as a foundation for program-to-program communication, independent of the types of processors in which these programs run.

The term **duplex** used in this text implies logical duplex. Duplex transmission allows two tasks to run simultaneously; one to receive input from the host and the other to send to the host. This term is sometimes referred to as full duplex.

The term **inbound** used in this text implies operation flow from the host to the Series/1.

The term **outbound** used in this text implies operation flow from the Series/1 to the host.

The term **SDLC** used in this text refers to the Series/1 SDLC support available with the SNA program product. This SDLC support provides exclusive use of the link to SNA; it does not allow SNA to use shared link support. Neither primary SNA (PSNA) nor APPC can share the SDLC support available with the SNA product.

The term **shared SDLC** used in this text refers to the Series/1 SDLC support that is available with the EDX Version 6 program product. SNA, PSNA, and APPC can simultaneously use the shared SDLC support. The shared SDLC support allows SNA to use the shared link support with APPC or another SNA PU.

The term **DLM** used in this text applies to the data link manager. The term **extended DLM** is the data link manager support that controls the interface between SNA and shared SDLC.

The term **PSNA** used in this text applies to the support for a Primary SNA network. Primary SNA is a partial implementation of the functions as defined by the Systems Network Architecture for PU type 4 and 5 devices.

The term **shared link** applies to the support in the shared SDLC available with the EDX Version 6 program product that allows more than one PU to share an SDLC link. The shared link support allows one or more SNA PU(s) and one or more APPC PU(s) (when APPC is running in a secondary environment) to share one set of hardware to the host; namely, 1 card, 1 cable, 1 set of modems, 1 line. Besides allowing SNA to share a link with APPC, the shared link support allows a SNA PU to share the link with one or more other SNA PU(s). Shared link support allows you to decrease your line costs.

The term **LU** used in this text refers to the logical unit.

The term **PU** used in this text refers to the physical unit.

Related Publications

The following list states the complete titles of related publications. The related publication titles referred to later in this book do not include the company, system, or version designations.

For a Version 6 operating system

- *IBM Series/1 Event Driven Executive Network Definition Utility Guide*, SC34-0764
- *IBM Series/1 Event Driven Executive Primary Systems Network Architecture Programming Guide*, SC34-0762, and Supplement SD34-2500
- *Advanced Program-to-Program Communication Programming Guide and Reference*, SC34-0960
- *IBM Series/1 Event Driven Executive Operator Commands and Utilities Reference*, SC34-0940
- *IBM Series/1 Event Driven Executive Language Reference*, SC34-0937
- *IBM Series/1 Event Driven Executive Messages and Codes*, SC34-0939
- *IBM Series/1 Event Driven Executive Installation and System Generation Guide*, SC34-0936
- *IBM Series/1 Event Driven Executive Operation Guide*, SC34-0944
- *IBM Series/1 Event Driven Executive Language Programming Guide*, SC34-0943
- *IBM Series/1 Event Driven Executive Customization Guide*, SC34-0942
- *IBM Series/1 Event Driven Executive Problem Determination Guide*, SC34-0941
- *IBM Series/1 Event Driven Executive Installation and Operation Guide for the Series/1 System Unit Programming RPQ P82910*, SC34-0934

For a Version 5 operating system

- *IBM Series/1 Event Driven Executive Primary Systems Network Architecture Programming Guide*, SC34-0762, and Technical Newsletter SN34-0961
- *IBM Series/1 Event Driven Executive Operator Commands and Utilities Reference*, SC34-0644
- *IBM Series/1 Event Driven Executive Language Reference*, SC34-0643
- *IBM Series/1 Event Driven Executive Messages and Codes*, SC34-0636
- *IBM Series/1 Event Driven Executive Installation and System Generation Guide*, SC34-0646
- *IBM Series/1 Event Driven Executive Operation Guide*, SC34-0642
- *IBM Series/1 Event Driven Executive Language Programming Guide* SC34-0637
- *IBM Series/1 Event Driven Executive Customization Guide*, SC34-0635
- *IBM Series/1 Event Driven Executive Problem Determination Guide*, SC34-0639
- *IBM Series/1 Event Driven Executive Installation and Operation Guide for the Series/1 System Unit Programming RPQ P82910*, SC34-0634

Also recommended are the following:

- *IBM Series/1 Event Driven Executive Commercial Applications Development Guide*, SC34-0381
- *Systems Network Architecture Format and Protocol Reference Manual: Architecture Logic*, SC30-3112
- *Systems Network Architecture Formats*, GA27-3136
- *Information Management System/Virtual Storage (IMS/VS):*
  - *Programming Guide for Remote SNA Systems*, SH20-9054
  - *Message Format Service (MFS) User's Guide*, SH20-9053
  - *Installation Guide*, SH20-9081
- *Customer Information Control System/Virtual Storage (CICS/VS);*
  - *IBM 3790 Guide*, SC33-0075
  - *System/Application Design Guide*, SC33-0068
- *OS/VS2 System Programming Library: JES3,,* GC28-0608
- *OS/VS2 MVS JES3 Commands*, GC23-0008
- *OS/VS2 System Programming Library: JES2*, GC23-0002
- *OS/VS2 MVS JES2 Commands*, GC34-0007
- *OS/VS1 System Generation Reference*, GC26-3791
- *OS/VS1 RES System Programmer's Guide*, GC28-6878
- *OS/VS1 RES Workstation User Guide*, GC28-6879
- *OS/VS Message Library: VS1 RES RTAM and Account Messages*, GC38-1010
- *OS/VS1 RES Operator's Library Reference*, GC38-0110

- *DOS/VSE System Generation*, GC33-5377
- *DOS/VSE Entry User's Guide*, GC33-6047
- *System/370 Operator's Reference Guide*, SR20-4460
- *VSE/POWER Installation and Operations Guide*, SH12-5329
- *VSE/POWER Messages*, SH12-5520
- *ACF for NCP and System Support Programs Installation and Resource Definition*, SC30-3167
- *IBM Implementation of X-21 Interface General Information Manual*, GA27-3287
- *IBM Series/1 Communication Theory Diagrams*, SY34-0059
- *VTAM Installation and Resource Definition*, SC27-0610
- *Synchronous Data Link Control Communications Feature Description*, GA34-0245.

Utilities Examples

In this book, the examples of using utilities are illustrative, not exact. For instance, you might see the following screen example:

```
>$L  $JOBUTIL  BLDNET1,EDX003
LOADING $JOBUTIL      nnP, hh:mm:ss, LP= xxxx, PART= yy
```

Symbol	Explanation
nnP	Indicates that the utility is nn number of pages long (256 bytes equals one page)
hh:mm:ss	Indicates the time in hours, minutes and seconds
LP = xxxx	Indicates the load point of the utility
Part = yy	Indicates the partition where the utility is loaded.

Contacting IBM about Problems

You can inform IBM of any inaccuracies or problems you find with this book by completing and mailing the ***Reader's Comment Form*** provided in the back of the book.

If you have a problem with the Series/1 Systems Network Architecture or Systems Network Architecture Remote Job Entry products, refer to the *IBM Series/1 Software Service Guide*, GC34-0099.

Chapter 1. Introduction

A data communications architecture defines the rules for dialogue between diverse programs and hardware products using telecommunications linkages. Systems Network Architecture (SNA), the communication architecture used by many IBM programs and communication products, provides a unified structure and protocol for communications network operation. The Event Driven Executive Systems Network Architecture provides a subset of the total SNA architecture for communications with System/370 host processors. The Series/1 SNA[1] subset consists of the following:

- A Series/1 defined as a cluster controller (PU type 2) with multiple LUs. The Series/1 contains Event Driven Executive Version 5.2 or Version 6, SNA support, and user programs. Each attachment card in the Series/1 functions as a separate EXIO device.
- A System/370 defined as a host (PU type 5) containing a virtual operating system, VTAM or TCAM access method, and Data Base/Data Communication (DB/DC) subsystem (IMS/VS, CICS/VS) application programs.
- An IBM 3705 Communications Controller (PU type 4) connecting the Series/1 with the System/370. The Network Control Program (NCP) resides in the communications controller and controls the operation of the controller.

The SNA functions supported for Series/1 also include the following:

- SNA function management profile 3 or 4
- SNA transmission subsystem profile 3 or 4
- SNA path control.

The subset of Series/1 SNA support allows your Event Driven Executive application program to communicate with a System/370 Data Base/Data Communication (DB/DC) subsystem (IMS/VS,CICS/VS) application program.

In an Information Management System/Virtual Storage (IMS/VS) environment, Series/1 SNA support enables the Series/1 to function as a 3790-type controller and conform to the formats and protocols defined by IMS/VS for a type "P" secondary LU.

In a Customer Information Control System/Virtual Storage (CICS/VS) environment, Series/1 SNA support enables the Series/1 to function as a 3790-type controller and conform to the formats and protocols defined by CICS/VS for a "full function" secondary LU.

The Series/1 SNA application programmer is able to emulate the CICS 3650 Pipeline Logical Unit when using duplex. The Series/1 SNA bracket protocol is compatible with the protocol used for the Pipeline Logical Unit.

See Figure 1-1 on page 1-4 for an illustration of the architecture and software in a Series/1 SNA network.

[1] SNA and Series/1 SNA used throughout this document refer to Event Driven Executive Systems Network Architecture.

Communication with a System/370 application using the Series/1 SNA support requires that you:

- Establish a session
- Send and receive messages, status, or error information
- Terminate the session.

An application programmer writes a program using the application program instruction set (EDX). Through the application program instruction set, a program can use the SNA product to communicate with the SNA network.

The interface to the SNA environment provides for all of these requirements by an application program instruction set (Event Driven Language). This instruction set performs the SNA commands and controls the network for you, thus relieving you of having to have extensive knowledge of SNA. However, you do need to know the requirements of the host subsystem application program with which you are going to communicate.

SDLC Support Available

Synchronous Data Link Control (SDLC) provides the communications line discipline between the Series/1 and the 3705.

Two forms of SDLC support are provided:

- SDLC
- Shared SDLC.

The form of support you can use depends on the EDX supervisor you have installed and the functions you want to use. The SDLC support runs on an EDX Version 5.2 or 6 system but does not enable you to take advantage of some of the function available on the Version 6 system. The shared SDLC support runs on the Version 6 system but not on Version 5. For more information, see "SDLC Support" and "Shared SDLC Support" on page 1-3.

Determining Which SDLC Support to Use

If you plan to use SNA in an EDX Version 5.2 environment, then you must use the SDLC support available in the SNA product.

If you require X.21 switched support in an EDX Version 6 environment, then you must use the SDLC support available in the EDX SNA product.

If you plan to use SNA in an EDX Version 6 environment and do not require X.21 switched support, then use the shared SDLC support available in EDX Version 6.

SDLC Support

The **SDLC** is the SDLC support available in the SNA Version 2 program product. This SDLC supports the following:

- An SDLC attachment feature #2090 with a RS-232-C interface either switched or nonswitched at line speeds up to 19.2Kb/sec.
- A synchronous communications single-line control/high speed attachment feature #2080. Using an X.21 link, either switched or nonswitched, the feature #2080 is supported at line speeds up to 19.2Kb/sec. Using a V.35 interface with a non-switched point-to-point link, the feature #2080 is supported at line speeds up to 56Kb/sec.

Note: If you are using SDLC support and you want to use the feature #2080 card with a V.35 interface to run a 56Kb/sec line speed, you must include X.21 circuit switched support in the EDX supervisor. Moreover, when using SDLC, you must relink your $SNA program with the SDLC support to include X.21 support. Check the SNA program directory for the procedures for using SNA with X.21.

Shared SDLC Support

The **shared SDLC** support is the SDLC available in the EDX Version 6 program product.

The shared SDLC supports the following:

- An SDLC attachment feature #2090 with a RS-232-C interface, either switched or nonswitched, at line speeds up to 19.2Kb/sec.
- A synchronous communications single-line control attachment, feature #2080. Using an X.21 link, nonswitched, the feature #2080 is supported at line speeds up to 48Kb/sec. Using a V.35 interface with a nonswitched link, the feature #2080 is supported at line speeds up to 56Kb/sec.

In addition, the shared SDLC support contains shared link support. Shared link support allows both SNA Version 2.1 and APPC to "share" one line between the host and the Series/1, decreasing your line costs. The shared link support uses the feature #2090 or #2080 cards. Shared link support requires a user to specify multiple physical units to be defined on the LINE macro in the host NCP generation.

The shared SDLC support requires the Network Definition Utility program product to define the shared SDLC characteristics. The Network Definition Utility provides a set of easy-to-use menus to define the configuration records for shared SDLC support and the synchronous data link control (SDLC) devices in the network. Using the utility, you can create records that describe the characteristics of the devices in the network. The Network Definition Utility allows you to dynamically change the shared SDLC configuration.

Note: With shared SDLC support, you do not have to set your modem eliminators to provide delay periods of at least 8 to 16 milliseconds for line turnaround. This increases the line utilization and throughput for some applications.

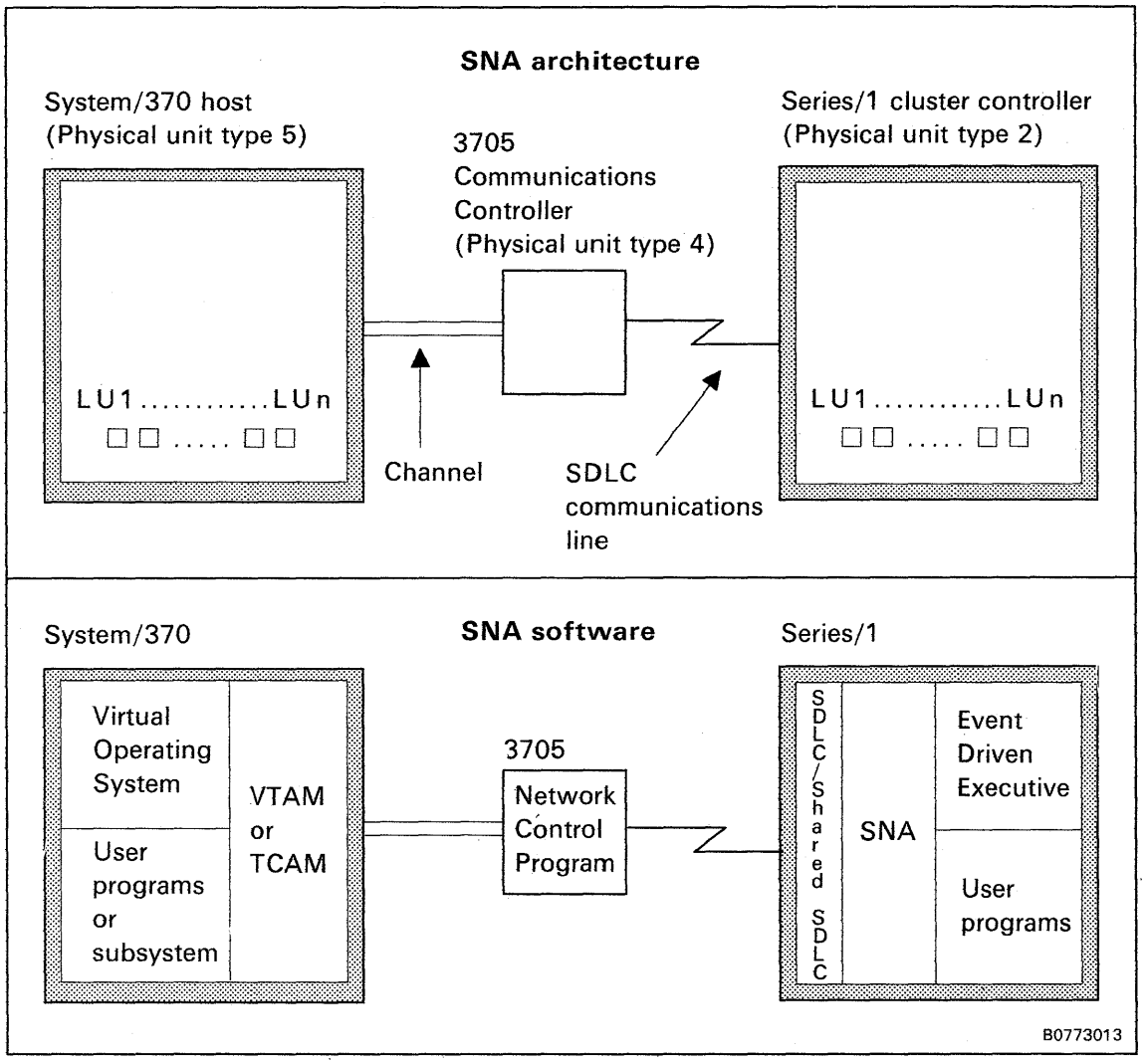


Figure 1-1. SNA Architecture and Software

SNA Main Program ($SNA)

SNA consists of a main program, $SNA, and up to four subprograms, $NET1 – $NET4. Each $NETx program (where x is a number from 1 to 4) represents an SNA PU; $NETx loads only when the PU activates.

$SNA contains the SNA support. It also contains the tables needed to manage four possible PUs. $SNA's functions include the following:

- Obtaining storage needed by the SNA network to manage the PUs
- Loading the PU programs when SNA activates the PUs
- Reloading the PU programs if necessary
- Providing the SNA support for the SNA network
- Deactivating a PU program if necessary
- Deactivating the entire Series/1 SNA network.

If you are using the SDLC available in the SNA product, then $SNA also contains the SDLC support for the SNA network.

If you are using the shared SDLC support in the EDX Version 6 program product, then $SNA does not contain the SDLC support. The $SNA program will contain the extended DLM support for the SNA network.

PU Program (NETx)

$NET1 – $NET4 represent separate PUs within the network. Each contains the following:

- All control blocks used by SNA, except the PU management control blocks used by $SNA
- All network definitions for the PU as defined in $SNADEFx (where x is a number from 1 to 4)
- All stacks defined for the PU
- All network and session buffers defined for the PU
- The SNA error routine program
- The SNA inbound message handler task control blocks for the PU.

If you are using the SDLC available in the SNA product, then $NETx also contains the SDLC control blocks and the SDLC task control block.

If you are using the shared SDLC support in the EDX product, then $NETx also contains the extended DLM control blocks and the extended DLM support task control block.

Overview of SNA Instructions

SNA provides seven application program instructions, as follows:

NETPACT activates a specific PU

NETHOST builds a host ID data list

NETINIT establishes an SNA LU-to-LU session

NETPUT sends a message during an SNA session

NETGET receives a message during an SNA session

NETCTL sends or receives status or error information to/from the host subsystem

NETTERM ends an SNA LU-to-LU session.

Starting a PU

You must activate the PU before you start any session using LUs defined for that PU. You can activate the PU on the Series/1 by either auto-activating the PU when loading $SNA (using the SNAINIT data set), activating the PU using the PUACT attention list command, or from your application using the NETPACT instruction.

Activating a PU on the Series/1 does the following:

- Loads all the control blocks needed by the PU
- Starts the SNA message handler for the PU
- Prepares the PU for user instructions.

If you are using SDLC, activating a PU on the Series/1 starts the SDLC support for the PU.

If you are using shared SDLC, activating a PU on the Series/1 starts the extended DLM support for the PU.

The PU and LUs must also be active on the host system, but you can activate them either before or after the PU on the Series/1 activates. Activating the PU on the Series/1 also prepares for use all the LUs defined for that PU. However, before you can establish a session on an LU, both the owning PU and the LU must be active on the host.

A PU remains active until deactivated by a Series/1 attention list command (PUDACT, SNADACT), until deactivated by the host (SDLC DISCONNECT command), or until SNA detects a hardware error on the SDLC link. If you define reactivation, the PU automatically reactivates if either the host deactivates the PU or a hardware error occurs.

Establishing a Session

You must establish a connecting path between an application program located on the System/370 host processor and a Series/1 application program so that the two programs can communicate with each other. In addition, the participating programs must agree to the rules and protocols for the use of the established path. For the Series/1, the NETHOST instruction defines the specific host application program; the NETINIT instruction establishes the path for program-to-program communications.

If the rules and protocols described by the NETINIT instruction are in agreement with the host application, NETINIT establishes a *session*: the participants, called *session partners*, can now communicate. A session continues as long as it is needed by the session partners or until it ends because of a system or network condition.

Sending Messages

Session partners transfer data in units called *messages*. The message size depends on the requirements of the host and Series/1 applications. Messages are grouped into units of work called *transactions*. A transaction can comprise a single message or several messages. The session partners define the beginning and end of each transaction on the session. Only one transaction can be in process at any time during a session.

To send messages to the host application, use the NETPUT instruction. This instruction provides options to verify messages and select function management headers.

Function Management Headers

Messages exchanged in a session can contain *function management headers* (FMH). The Series/1 application develops the FMH and appends it to the beginning of the message text. The FMH, if used, must subscribe to the exact format and content requirements of the host application. You must indicate the presence of the FMH at the beginning of the message by coding the FMH parameter on NETPUT.

The Right-to-Send

Within a transaction, session partners must take turns sending messages. To send a message, a session partner must have the *right-to-send.* The session partner that has the right-to-send can grant it to the other session partner. The session partner starting the transaction begins with the right-to-send.

When no transaction is active in the session, both session partners have the right-to-send. If both partners attempt to send at the same time, the Series/1 transaction starts, and SNA rejects the host application transaction. (The host may retry the rejected transaction later.)

Message Verification

You can optionally request message verification on NETPUT operations. A NETPUT operation cannot complete until the application receives confirmation of the acceptance (or rejection) of the message.

For sessions supporting resynchronization, Series/1 SNA always waits for the verification of the current message before completing the NETPUT operation; therefore, the host can reject only the current message.

Message Resynchronization

Message resynchronization helps to detect lost messages when a session fails and is reestablished by a subsequent NETINIT instruction. A NETINIT parameter requests message resynchronization (RESYNC=YES or INIT) and remains in effect until the end of the session.

An area defined by NETINIT parameters preserves values that relate to the message exchange; SNA uses these when restarting a session. A NETINIT return code reflects a lost message. The host subsystem initiates resynchronization when it restarts the session. Similarly, the host system accumulates data for recovery when the Series/1 SNA support informs it of a lost message.

The Series/1 SNA application program is responsible for the recovery of lost messages that were sent to the host.

With resynchronization, every message that is sent by the Series/1 application does not complete until the host subsystem responds to it; and the Series/1 gives the right-to-send to the host for every message sent.

During a session, all messages transmitted between the host subsystem and the Series/1 SNA application are numbered. The host assigns numbers for messages received; Series/1 SNA support assigns numbers to messages sent.

For sessions established with the resynchronization support, the number of the last message sent in each direction is saved for resynchronization. The Series/1 SNA application does not need to be concerned with message numbers, as the handling of the message numbers is internal to the Series/1 SNA support.

For sessions without resynchronization support, the message numbers of messages sent to the host are optionally returned to the SNA application on NETPUT operations. You can use these numbers to correlate negative acknowledgments from the host to messages sent by your application.

Receiving Messages

To receive messages from the host, you use the NETGET instruction. When the receive operation is issued, the host application is granted the right-to-send (if not already in possession of it). The NETGET instruction completes when the end-of-message is received or the buffer to receive the message is filled.

Upon completion of the receive operation, the return code informs you of the results of the operation. Unless it receives the right-to-send or end-of-transaction, the host has more messages to send. To receive the messages, issue additional NETGET instructions. *Attention events* serve to alert a session of a message arrival from the host.

Function Management Headers on Receive Operations

Series/1 SNA support recognizes received FMH messages and notifies the application with a return code signifying that a FMH message is in the buffer. The FMH message is not evaluated by SNA and is passed to the application along with the message data.

Message Verification

The host application can, in sending a message, request acknowledgment of the Series/1 receipt of the message; the return code indicates this. The operation following the NETGET instruction determines acknowledgment of such messages (acceptance or rejection). You must use the Series/1 SNA control instruction, NETCTL, for all messages that are rejected. Any operation other than a rejection implies acceptance of the message. In any message flow, all messages received prior to a rejected message are accepted.

Controlling Message Exchange

The NETCTL instruction receives or sends session status. To receive status, you should issue a NETCTL instruction after an SNA instruction operation where the return code indicates "status available." The Series/1 SNA support prevents the use of any other instruction (except NETTERM) until it retrieves the status information.

The status information *received* from the host application on a NETCTL operation is one of the following:

- Request for the right-to-send
- Identity of a rejected message and the reason for its rejection
- Cancel of the message being received
- Request of suspension or resumption of message traffic sent to the host
- Request for session termination
- A status message.

Note: To receive status, you can issue the NETCTL instruction regardless of which session partner has the right-to-send.

The Series/1 application uses NETCTL to *send* status to the host application in a similar operation.

NETCTL informs the host application of specific message processing needs of the Series/1 application, such as:

- Suspend sending messages — Used to temporarily stop the host from sending messages.
- Resume sending messages — Used to allow the host to start sending messages after being suspended.
- Request the right-to-send from the host — The Series/1 application finds that while receiving messages, it needs to send a message.
- Cancel a message — A message that was partially sent by the Series/1 is to be discarded by the host. Only messages where the end of message is not indicated can be cancelled.

NETCTL also acknowledges messages. These acknowledgments are message acceptance or message rejection. Message acceptance (required only when the host specifically requests acknowledgment to a message) and rejection requires using NETCTL as the next Series/1 SNA operation following the NETGET operation. In rejecting the message, the Series/1 application places the reason for the rejection in the status area.

Ending a Session

The NETTERM instruction ends the LU-to-LU session and releases the resources used in the session. The session termination instruction is usually issued when both session partners agree that all transactions are complete (or that continuing dialogue serves no purpose). Since a NETTERM can be issued at any time, your Series/1 application can end a session without agreement from the host application, if it is appropriate to do so.

Session termination can also occur if the host abnormally ends the session. This is an unconditional termination and is usually caused by the host's detection of a severe error condition. Following an abnormal termination, you should issue NETTERM to free session resources.

If any sessions were established, you should not use $C to cancel your application without first issuing a NETTERM to free any active sessions. $C does not free all the SNA resources, and unpredictable results may occur. Also, you should not end your program (with the EDL ENDPROG instruction) until you issue a NETTERM for each active session.

Session Reset

Session reset occurs when the host detects an error that does not warrant session termination, but does require some session reinitialization. Under this circumstance, the host sends the CLEAR message, followed by Start Data Traffic (SDT) to reset the session. To end the session, issue a NETTERM. To continue a session, issue a NETGET, NETPUT, or NETCTL.

Note: After a session reset occurs, pending messages are cancelled.

Coding and Executing Your Application

This section includes general information to help you develop and execute your Series/1 SNA application.

Return Codes

In coding your application, you should understand the requirements of the host application and the codes that can return to your application from the Series/1 SNA support.

In many cases, the return code dictates the course of action to be taken by your application. To maintain proper session protocol, you must check the return codes in your application. The return codes are described at the end of each instruction description.

The Event Driven Executive Series/1 SNA instructions return to your SNA application program a return code upon completion of the instruction. These return codes appear in the first word of the task control block ($TCBCO) of the task that issued the instruction.

Return codes enable you to determine the success or failure of an operation you requested. When processing return codes, you should first check for return codes less than −1. Return codes less than −1 specify the operation did *not* complete successfully, and you should take appropriate action.

If the return code is −1, the operation completed successfully.

If the return code is positive, the operation was successful and you should check the return code bit settings to determine additional information concerning the operation.

Additional or "extended" error information returns, when appropriate, to your application if you specify the ERRCODE parameter of the NETINIT instruction. Refer to Chapter 11, "Extended Error Information" for the extended error return code information and meaning. For many errors, the extended error information is the only way you can determine the actual cause of a failure.

Priorities

The highest priority on your TASK statement must not exceed 60. In this instance, a task with a priority of 10 has a higher priority than a task with a priority of 60. So, for example, a task with priority of 50 would be invalid and might cause unpredictable errors.

SNA Task Error Exit

A task error exit routine ($SNAERR) is included in each SNA support program ($NETx). This routine receives control if a hardware error or program exception occurs during one of the following instances:

- While the SNA inbound message handler task is executing
- In the case of SDLC support, while the SDLC task is executing
- In the case of shared SDLC support, while the extended DLM task support is executing.

Upon encountering such an error, the $SNAERR routine displays a message informing you that the PU abnormally terminated and displays the address of the area from which you should take a storage dump.

You can replace the $SNAERR task error exit routine with a user-written program if you want an application-dependent error routine. For additional information on task error exit handling, refer to the *Customization Guide*.

Using the Parameter-Naming Operands

You can supply parameter values during execution by using the Px parameter-naming operand. The following Event Driven Executive SNA instructions allow the Px= operand:

- NETPUT
- NETGET
- NETCTL
- NETINIT
- NETPACT
- NETTERM.

The parameter naming operands enable you to supply data to instructions in your program without having to define that data with a DATA, DC, or TEXT statement.

The Px= operands correspond to other operands in the instruction syntax. P1= represents the first operand in an instruction, P2= represents the second operand, P3= represents the third operand, and so on. The number of parameter-naming operands allowed within each instruction varies.

Each instruction describes the actual parameter to Px relationships.

To use a Px= operand, you must first code it with a label. The label refers to a storage location within the instruction. The system treats the label as the parameter of the operand to which the Px= operand refers. Once you assign a label to the Px= operand, you can use that label in other instructions in your program.

For the exact description of Px operands in different instructions, see the operand description for Px in:

- Chapter 4, "Network Operation" for NETPACT
- Chapter 5, "Activating a Session - NETHOST and NETINIT"
- Chapter 6, "Sending Messages - NETPUT"
- Chapter 7, "Receiving Messages - NETGET"
- Chapter 8, "Controlling Message Exchange - NETCTL"
- Chapter 9, "Terminating a Session - NETTERM."

Synchronous Operation

All SNA instructions are synchronous in their operation. That is, the instruction must complete prior to returning control to your application.

Loading $SNA from Your Application

If you load $SNA from your application, you must specify the PARMNAME parameter on your load instruction. This list must contain eight words of zeros, unless you want to override the SDLC station ID defined for any of the PUs by the STAXID parameter on the SNAPU statements. See Chapter 4, "Network Operation" for a further description of this field.

Linking Your Application

When you link your application, you must include $NETCMD and/or $NETPACT (supplied with the distribution material) in your link control statements. $NETCMD and $NETPACT are the connection between your application and the Series/1 SNA support. The Series/1 SNA instructions when executed generate an inline "user" call to $NETCMD or $NETPACT. You must also include the autocall option of $EDXLINK in your link control statements when linking your Series/1 SNA applications. For example:

```
AUTOCALL $AUTO,ASMLIB
```

You can issue all operations in different tasks than the task that issued the NETINIT, but the task must be running under the same program that is linked to $NETCMD or $NETPACT; otherwise, an error occurs.

NETINIT, NETPUT, NETGET, NETCTL, and NETTERM all require $NETCMD, and NETPACT requires $NETPACT. If your application does not issue NETINIT, NETPUT, NETGET, NETCTL, or NETTERM, you need include only $NETPACT. If you do not issue any NETPACT instructions, you need include only $NETCMD.

Illustrations of SNA Instruction Sequences

The following two coding illustrations show the sequences of Series/1 SNA instructions used for a session with a host subsystem. Subsequent chapters describe these instructions in more detail.

Illustration 1

Figure 1-2 on page 1-13 shows how you can use the Series/1 SNA instructions to perform the following:

1. Establish a session with a host using the NETINIT instruction.
2. Send a transaction ID to the host using a NETPUT instruction.
3. Issue a NETGET or a series of NETGET instructions (until the end-of-transaction is received) to receive data from the host.
4. Issue a NETCTL to send status to the host.
5. End the session with the host by issuing a NETTERM instruction.

```
         NETINIT  LU=NETLU,HOSTID=SNAID,ACQUIRE=YES,                C
                ERRCODE=SAVERC,RESYNC=NO
```

(process return code)

```
         NETPUT   LU=NETLU,BUFF=OUTBUFF,BYTES=BYTECNT,              C
                INVITE=YES,LAST=YES
```

(process return code)

```
GETIT    NETGET   LU=NETLU,BUFF=INBUFF,BYTES=INBLEN,               C
                RECLEN=COUNT
```

(process return code; if no errors have occurred, continue looping on GETIT until the return code indicates end-of-transaction)

```
         NETCTL   LU=NETLU,TYPE=LUSTAT,BUFF=STATUS
```

(process return code)

```
         NETTERM  LU=NETLU
```

```
*****************************************************
NETLU    DATA     F'1'
SAVERC   DATA     F'4'
SNAID    NETHOST  ISAPPID=IMS,ISMODE=INQUIRY
OUTBUFF  DATA     CL80'TRANID'
BYTECNT  DATA     F'80'
INBUFF   DATA     XL'80'
INBLEN   DATA     F'80'
COUNT    DATA     F
STATUS   DATA     XL6
         ENDPROG
         END
```

Figure 1-2. Illustration of a Series/1-Initiated Transaction

Illustration 2

Figure 1-3 on page 1-15 shows the sequence of the Series/1 instructions that you can use to wait for and receive data transmitted from the host. The coding illustration shows the following:

1. Starting the PU for the LU.
2. Establishing a session with a host using the NETINIT instruction.
3. Waiting for the attention event to be posted signaling that the Series/1 SNA received data from the host.
4. Receiving the data by issuing the NETGET instruction. If Bid was received (−26 return code) by the Series/1 application, you should issue another NETGET to receive the data and reset the attention event. Continue to issue WAITs and NETGETs until end-of-transaction is received.
5. Sending status to the host by issuing a NETCTL TYPE=LUSTAT.
6. Ending the session by issuing a NETTERM instruction.

```
         NETPACT  PU=PUNUM

       (process return code)

         NETINIT  LU=NETLU,HOSTID=SNAID,ACQUIRE=YES,                C
               ERRCODE=SAVERC,RESYNC=NO,ATTNEV=SNAECB

     (process return code)
ATTNPROC WAIT     SNAECB
RSETPROC RESET    SNAECB

GETIT    NETGET   LU=NETLU,BUFF=INBUFF,BYTES=INBLEN,               C
               RECLEN=COUNT

       (process return code; if Bid received (-26 return code),
       perform GETIT to receive data, perform RSETPROC to reset
       attention event. If end-of-transaction was not indicated
       by return code and no errors have occurred, go back to
       ATTNPROC to wait for more data)
            •
            •
            •
         NETCTL   LU=NETLU,TYPE=LUSTAT,BUFF=STATUS

        (process return code)
            •
            •
            •
         NETTERM  LU=NETLU
            •
            •
            •
*****************************************************
PUNUM    DATA     F'1'
NETLU    DATA     F'1'
SAVERC   DATA     F'4'
SNAID    NETHOST  ISAPPID=IMS,ISMODE=INQUIRY
SNAECB   ECB      0
INBUFF   DATA     XL'80'
INBLEN   DATA     F'80'
COUNT    DATA     F
STATUS   DATA     XL6
         ENDPROG
         END
```

Figure 1-3. Illustration of a Host-Initiated Transaction

SDLC/SNA Line Trace

SNA uses the EDX communications line trace facility ($LINTRC) to capture and report data traffic flowing across SDLC lines. When a problem occurs in SNA, whether you are using SDLC or shared SDLC, or in an application, you can use the line trace facility to decrease the time you use diagnosing the problem. The EDX communications line trace is transparent to SNA and the application. However, since capturing the line data increases overhead, your applications may need more time to complete.

The EDX line trace utility ($LINTRC) can capture and report data traffic on any SDLC lines attached to the Series/1 processor where the utility is running. You can use the facility to trace several lines concurrently. The utility ($LINTRC) captures data to a uniquely named disk data set for each line. Later, you can analyze this data by using the format utility ($LINUT1). Because these data sets are standard EDX data sets, you can copy them from Series/1 to Series/1, and a network operator on a remote system can view them.

To start tracing data for a particular line, you must know the SDLC device address associated with that line. You then load the $LINTRC utility. When loading the utility, you specify the trace data set that will contain the data and the SDLC device address for the line you are tracing. The $LINTRC utility also enables you to set options. For instance, you can set the actual size of the line trace data set and determine how much data the utility should save. You can also tell the utility what to do if the line trace data set fills with data (wrap or no wrap). For more information about using the $LINTRC utility, refer to the *Operator Commands and Utilities Reference.*

An option of the $LINTRC utility enables you to specify how much of the data should be saved in the trace data set. If you have an application which is sending very large messages, but the control information for the message is always located in the beginning of the record, you might not need to save the entire message to analyze the problem. For instance, if you are experiencing SNA or SDLC problems, you usually need to capture only the first 18 bytes of the data. These bytes usually provide enough clues for you to find the problem area. These 18 bytes contain information, as follows:

Byte	Description
1	SDLC station address
2	SDLC control byte
3-8	SNA transmission header
9-11	SNA request/response header
12-18	SNA request code and sense data.

Some SNA problems might require you to capture more than the recommended 18 bytes of data (for example, if you have a BIND specification error); however, you can solve most problems by using the clues in those 18 captured bytes. If you plan to analyze SNA or SDLC data, you should have copies of both the *SNA Format and Protocol Reference Manual: Architecture Logic* and the *SNA Reference Summary.*

Once the data is captured, you can use the $LINUT1 utility to format and view the captured data. This utility interrogates and interprets the captured data and displays the data in either unformatted or formatted form. If you are not very familiar with SNA protocols, you might want to use the utility to view the data in formatted form. In formatted form, the SNA headers are interpreted, and displayed in an

easy-to-read format. If you are familiar with SNA transmission and request/response headers, then you might want to display the data in unformatted form. Unformatted form gives you a hexadecimal and EBCDIC dump of the data. You can view unformatted data faster than formatted data.

Along with the captured data, the trace facility captures the device commands between SDLC software and the SDLC device. You can set the $LINUT1 utility to display or ignore captured device information. The line trace facility captures the following device information:

- Immediate Device Control Block (IDCB) for each I/O
- Optional Device Control Block(s) (DCBs) for each I/O
- I/O condition code for each I/O
- Interrupt Condition Code (ICC) for each I/O
- Interrupt Information Byte (IIB) or Information Status Byte (ISB) associated with each interrupt
- Residual Status Block (RSB) for each DCB.

Notes:

1. The EDX communications line trace facility is only supported by EDX Version 5.2 and subsequent versions.
2. When the communications line trace facility is capturing data for an SDLC line, a decrease in performance will occur.

It is strongly recommended that any APARs submitted to IBM include a trace of the line when the problem occurred. If a system dump is taken, the line trace submitted should correspond to the system dump.

Chapter 2. Installing and Defining the SNA Network for Use with SDLC

The SDLC support discussed in this chapter is that SDLC support available in the SNA Version 2 program product.

This chapter tells you how to install SNA for use with SDLC on your system, how to prepare your EDX system for SNA, and how to define your SNA network for use with SDLC. The chapter contains the SDLC return codes, storage requirements, and log record format. Refer to the EDX SNA Program Directory for a more detailed discussion and step-by-step instructions for installing SNA and defining SNA to EDX.

Note: If you plan to use the shared SDLC support that is available in the EDX Version 6 program product, you should see Chapter 3, "Installing and Defining the SNA Network for Use with Shared SDLC."

Installing SNA

You can use SNA Version 2.1 SDLC support with either an EDX Version 5.2 or an EDX Version 6 system. The Version of EDX you use determines which procedure you use to install SNA. If you are using a Version 5.2 operating system, see "Installing on an EDX Version 5.2 System." If you are using a Version 6 system, see "Installing on an EDX Version 6 System" on page 2-2.

Installing on an EDX Version 5.2 System

To install Version 2.1 on an EDX Version 5.2 system, you must run $XX9UT1 to install all needed modules. You must explicitly request copying of the EDX SNA object modules if desired.

Note: Unlike Version 1.2 of SNA, the EDX SNA object modules (except the ones copied automatically) are **not** required to define and configure your SNA system.

If you leave the object modules on diskette, you save disk space. Only the modules needed to assemble and build your SNA definitions and applications are automatically copied to your system.

If you want to install the object modules for the SNA main program on your system, enter **1** when the following screen appears:

```
        $XX9UT1 - INSTALLATION AID UTILITY
      EDX SYSTEMS NETWORK ARCHITECTURE (5719-XX9)
            *****   WARNING !    *****
EXISTING EDX SNA MODULES WILL BE DELETED
THIS UTILITY MUST NOT BE USED ON A V6.0 OR LATER VERSION SUPERVISOR

ENTER "END" TO ABORT EDX SNA INSTALLATION,
HIT ENTER TO CONTINUE ==>




ENTER:
1 - TO INSTALL SNA OBJECT MODULES ON VOLUME EDX003
2 - TO LEAVE THESE MODULES ON DISKETTE (SAVES DISK SPACE)
===>
```

$XX9UT1 copies all the SNA main program object modules from the diskette to your disk.

$XX9UT1 copies the prebuilt $SNASDLC program onto EDX002. $SNASDLC is the SNA main program that includes SNA with SDLC support. You must rename $SNASDLC to $SNA using $DISKUT1.

Aside from $SNASDLC, which $XX9UT1 copies automatically, you need copy additional modules only if you plan to relink the SNA main program. A relink is required only if you have customized SNA modules you want to include in the SNA main program.

Refer to the SNA program directory for more detailed installation procedures, and an example of changing the default installation specifications.

Warning:
If you do not copy the object modules from the diskette,
you must keep the diskette to apply APAR/PTF module replacements,
which require a relink of the SNA main program.
DO NOT DISCARD THE DISKETTE CONTAINING THESE MODULES.

Installing on an EDX Version 6 System

If you are installing SNA Version 2.1 with SDLC on an EDX Version 6 system, you must run $INSTAL to install the required SNA Version 2.1 modules. $INSTAL is an installation utility available with the EDX Version 6 product. $INSTAL maintains a history file of the current level of software on a system. $INSTAL can assist a user in determining the version, modification level, and APAR fixes of the software on each system.

If you do not use $INSTAL to edit the default installation specifications, the EDX SNA object modules are left on diskette and the $S1ASM macros are not copied. The only object modules automatically copied are the ones needed to assemble and build your SNA definitions and applications.

To change the installation specifications, you must use $INSTAL to edit the $$INXX9 control data set. By editing the $$INXX9 data set, you can specify that the EDX SNA object modules and/or the $S1ASM macros should be copied to disk.

To specify copying all SNA object modules to disk, use $INSTAL to edit $$INXX9 on volume XX9001. When prompted, change the $COPYUT1 control file name from "XX9001N" to "XX9001" on volume XX9001. To specify copying the $S1ASM macros, add the $COPYUT1 control file names "XX9002" and "XX9003" on volume EDX002.

$INSTAL copies the prebuilt $SNASDLC program onto EDX002. The $SNASDLC program contains the SNA Version 2.1 modules with SDLC support. You must rename the $SNASDLC program to $SNA using $DISKUT1.

You can choose your own history file name.

Refer to the SNA program directory for more detailed installation procedures, and an example of changing the default installation specifications.

Warning:
If you do not copy the object modules from the diskette,
you must keep the diskette to apply APAR/PTF module replacements,
which require a relink of the SNA main program.
DO NOT DISCARD THE DISKETTE CONTAINING THESE MODULES.

Defining SNA to EDX

To include SNA with SDLC in your EDX system, you must define one EXIO device for each PU, up to a maximum of four during the system generation process. Each PU requires a separate SDLC attachment card.

Do the following for an Event Driven Executive SNA system generation:

- Define the SNA EDL instructions to the EDX overlay assembler ($EDXASM) by adding the EXTLIB statement for $EDXLSNA to $EDXL if you use the overlay assembler to define your network or assemble your application programs.
- If you use $S1ASM instead of $EDXASM to define your network and assemble your applications, you must copy the macros from diskette onto your system. Otherwise, you do not need to include the macros in your system.

- Edit the $EDXDEF data set on volume ASMLIB to match the hardware required by your SNA and SDLC support. You should specify an EXIODEV statement for each SDLC device attachment card on your Series/1.
  - Each active PU requires a separate SDLC attachment card.
  - Each EXIODEV statement for an SDLC attachment must specify RSB=4 and MAXDCB as a value from 2 to 8.
  - The MAXDCB value must be the same as the value specified for the DCBNO parameter of the SNAPU statements.

  Notes:

  1. The address parameters on the EXIODEV and SNAPU statements must match.
  2. The value specified for the MAXOUT parameter of GROUP, LINE, or PU macros for the Network Control Program (NCP) generation must be 1 less than the value for MAXDCB, as in the following example:

```
EXIODEV    ADDRESS=0A,RSB=4,MAXDCB=2

SNAPU      ADDRESS=0A,DCBNO=2

PU         MAXOUT=1,...            (NCP macro)
```

- Edit the EDX $LNKCNTL data set on volume ASMLIB to include support for SNA and SDLC in your system. You must add an INCLUDE statement for the following modules in your system link statements:
  - **EDXTIMER** or **EDXTIMR2** for timer support
  - **FULLMSG** for full message data set support
  - **IOSEXIO** for EXIO device support.

  Notes:

  1. SNA requires timer support in the supervisor. SNA does **not** require a timer card.
  2. If timer support or full message support is not defined in the EDX supervisor, $SNA prints an error message and unloads.
  3. Include the $X21IA module if you are using X.21 circuit switched network support. Refer to the EDX SNA Version 2.1 Program Directory for more information on how to install your system when using X.21.

- Complete the system generation as described in the *IBM Series/1 Event Driven Executive Installation and System Generation Guide*.

Defining the SNA Network

You define your SNA network by defining PUs and LUs. In defining your network for use with SDLC, you must do the following:

1. Edit $SNADEFx
2. Edit LINKNETx
3. Run $JOBUTIL.

Editing $SNADEFx

When you edit $SNADEFx, you do the following:

- Edit the $SNADEFx data set. The $SNADEFx data set contains the definitions for the SNAPU and SNALU configuration statements.
- If you use only one PU, place your network definitions in $SNADEF1. $SNADEF1 is shipped with the product and contains default values for the PU and LU.
- If you use multiple PUs (up to a maximum of four), place each PU definition in $SNADEFx, where *x* is the number of the PU. Use $SNADEF2 for PU #2, $SNADEF3 for PU #3, and so on.

Running $JOBUTIL

After editing $SNADEFx and LINKNETx, you must build the PU using the $JOBUTIL control cards in BLDNETx (where x is the number, from 1 to 4, of the PU you are defining). $JOBUTIL creates programs named $NETx, where *x* is the number of the PU containing your PU definitions. SNA reserves the names $NET1 through $NET4, and you must not change them.

Note: $SNADEF1 contains configuration statements (SNAPU and SNALU) with default values for the PU and LU. The default values specified in $SNADEF1 comprise the default SNA network configuration defined in the SNA Program Directory. If your application requirements do not match the default values, you must tailor the network according to your specific application or network requirements by editing the configuration statements, SNAPU and SNALU.

SNAPU Configuration Statement

The SNAPU configuration statement generates a PU control block. The following is the syntax and parameter descriptions for the SNAPU configuration statement. (Before you redefine the default values for SNAPU parameters, review "SNAPU Parameter Considerations" on page 2-10.)

```
label      SNAPU     ADDRESS=,BUFNO=,BUFPOOL=,BUFSIZ=,CNCNAME=,          C
                     CNCTYPE=,DCBNO=,ENCODE=,FRTPOLL=,NETNAME=,          C
                     NORING=,PAD=,RATE=,RETRY=,SBUFNO=,STAXID=,          C
                     STKNUM=,THRESH=,TOCTS=,TODSR=,TODTR=,TOHLA=,        C
                     TONPR=,UNLOAD=

Required: label,
Defaults: ADDRESS=0A,BUFNO=2,BUFPOOL=YES,BUFSIZ=256,
          CNCNAME=X21RECyy,CNCTYPE=PP,DCBNO=2,ENCODE=NRZ,
          FRTPOLL=NO,NETNAME=BLANKS,NORING=NO,PAD=NO,
          RATE=FULL,RETRY=3, SBUFNO=0,STAXID=S00000,
          STKNUM=2,THRESH=1,TOCTS=29,TODSR=29,
          TODTR=29,TOHLA=29,TONPR=29,UNLOAD=NO
```

label (required)

The SNA network name. Recommended names are NETx, where *x* is the PU number (1 – 4).

ADDRESS
The physical device address (hexadecimal) of the SDLC card for the line that connects to this station. This address must match the address specified on the EXIODEV statement in $EDXDEF. Do not confuse this value with the secondary station address (which is jumpered on the SDLC attachment and used by the host when sending frames to the PU). The default value is 0A.

BUFNO
The number of SDLC buffers to be allocated as free pool buffers at network activation time. This is the number of buffers that will exist in addition to those allocated for each LU at network generation time. The minimum is 2, and the default is 2. This value must be at least 1 greater than the value for THRESH. For more information, see "SNA/SDLC Buffer Considerations" on page 2-15.

BUFPOOL
This operand specifies whether you use buffer pooling or session buffering for the PU.

YES (default)	You use buffer pooling. All buffers are shared between the LUs and the PU.
NO	You use session buffering. Buffers are not shared among LUs. The PU and the LUs each have separate buffer areas that are used exclusively by the LU or the PU.

BUFSIZ
The physical size of buffers used by SDLC, the PU, and the LUs. This buffer length is the length of the largest request/response unit (RU) to be sent or the largest RU segment received. This size determines the size of all buffers allocated at network generation time. The minimum is 50 (to allow receipt of a minimum length BIND), and the default is 256.

CNCNAME
This operand specifies a field containing the name of the connection record that the X.21 circuit switched network support uses when making a call. You can create a specific name for the connection record or leave CNCNAME blank, choosing the default name, which is X21RECyy. The name must consist of 1 – 8 alphanumeric characters. If you choose the default, do the following:

- Leave the CNCNAME parameter blank.
- Code the name field for the connection record as X21RECyy. In this case, yy is the device address in hexadecimal and is equivalent to the ADDRESS parameter.

Note: Use CNCNAME in conjunction with auto-call and direct-call. For more information on the X.21 connection record, see "Creating a Connection Record" on page 18-4.

CNCTYPE
Indicates the connection type and how the support is to manage the connection of the communications lines. The connection types are as follows:

PP (default)	Nonswitched, specify if point-to-point or multipoint
AA	Auto-answer
MA	Manual-answer
MC	Manual-call
AC	Auto-call
DC	Direct-call

Note: Auto-call and direct-call are valid only when using X.21 circuit switched network support. See Chapter 18, "X.21 Circuit Switched Network Support" on page 18-1 for more information on using X.21.

DCBNO

Specifies the maximum number of consecutive frames that you can receive or send. This value should be determined from the host system specifications for the station to be attached. The minimum is 2 (the default), and the maximum is 8.

The value specified for DCBNO must match the value for MAXDCB on the EXIODEV statement. DCBNO must be one greater than the value for the MAXOUT parameter of the GROUP, LINE, or PU macro for the Network Control Program generation.

ENCODE

This operand controls the mode of bit transmission.

NRZ (default) Set Nonreturn to Zero mode on the modem.

NRZI Set Nonreturn to Zero Inverted mode on the modem.

FRTPOLL

Specifies whether SDLC should time out if SNA does not receive a poll for this PU within a specified time interval when first activated. You set the interval in the TONPR parameter.

YES SDLC times out if the host does not poll within the specified time limit, even when first activated.

NO (default) SDLC waits forever for the first poll and uses the TONPR value on all other received items. You should specify FRTPOLL=NO if the Series/1 activates its PU and the host does not.

NETNAME

This operand specifies a field containing the network name (1 to 8 characters) of this PU. If not specified, the network name of this PU defaults to blanks.

NORING

Use this operand to specify if the modems provide the ring interrupts.

YES SNA must provide ring support, since the modem does not provide a ring interrupt.

NO (default) Ring support not required.

PAD

This operand controls the inclusion of pad characters.

YES Send the pad characters on the initial transmission.

NO (default) Do not send pad characters on the initial transmission.

RATE

Use this operand to control the speed rate of the modem.

FULL (default) Operate the modem at its normal speed.

HALF If the modem is equipped to recognize the "data signal rate select" line, operate the modem at one-half its normal speed.

Note: If you specify RATE=HALF, it is recommended that the TODSR operand must be at least 29.

RETRY
The number (0 – 255) of retries for each unsuccessful I/O request. The default is RETRY = 3.

SBUFNO
This operand specifies the number of buffers (0 – to the maximum available storage) allowed for concurrent send operations.

If buffer pooling is specified (BUFPOOL = YES), it is recommended that you enter a value for the SBUFNO parameter. If you specify a value other than 0, then you must also specify SENDBUF = 0 and RECVBUF = 0 on all of your SNALU macros. If you specify SBUFNO = 0, then your SNALU instructions must specify SENDBUF with a value of at least 2.

If session buffering is specified (BUFPOOL = NO), then you must specify SBUFNO = 0.

For either buffer pooling or session buffering, the minimum SBUFNO is 0 and the default is 0.

STAXID
The format and unique ID to be sent in response to an XID command to this station. This operand is for a switched line.

Ln Long format. Unique ID is *n*.

Sn Short format. Unique ID is *n*.

The value of *n* may be any number expressed as five hexadecimal digits. This ID should be unique among all Series/1 systems on the SNA network.

The format of the data in the 6-character field is as follows. The first character is expected to be an EBCDIC L or S. Any other value defaults to S. The next five characters are used as the unique ID (n). The default is the short format, STAXID = S00000.

STKNUM
Specifies the number of stacks the PU should reserve for use by the LUs. All stacks remain in a pool until you issue a request for an LU (NETGET, NETINIT, and so on). You must specify a value from 2 to 96 for STKNUM.

For half-duplex sessions, it is highly recommended that you specify a number based on the number of defined LUs plus one additional stack. For example, if you have defined two LUs, you would specify a total of three stacks.

For duplex sessions, it is recommended that you specify two stacks for each defined LU plus one additional stack. For example, if you have defined two LUs, you would specify a total of five stacks. The default is STKNUM = 2. For more information, see "Concurrent Requests" on page 2-18.

THRESH
Specifies the minimum number of buffers required to be available, when setting up for a receive operation, to be in receive ready state. The minimum value is 1 (default). The maximum is the value specified by the MAXDCB operand of the EXIODEV statement at system generation time.

TOCTS
Clear-to-send time-out. The period of time within which the modem should return **Clear To Send** to the adapter after requesting a transmit operation or an error is indicated (default is 29).

See "SDLC Device Time-out Considerations" on page 2-11.

TODSR
Data-set-ready time-out. The data set must become ready before this time-out occurs or an error is indicated. This time-out should include the time needed for modem equalization when changing rates (default is 29).

Note: If you specify RATE=HALF, then it is recommended that the TODSR operand be at least 29.

See "SDLC Device Time-out Considerations" on page 2-11.

TODTR
Data-terminal-ready time-out. The data set must become not ready before this time-out occurs or an error is indicated (default is 29).

See "SDLC Device Time-out Considerations" on page 2-11.

TOHLA
Hold-line-active time-out. The maximum time allowed for the adapter to prepare for the next transmit operation in a chained transmit sequence (default is 29).

See "SDLC Device Time-out Considerations" on page 2-11.

TONPR
The time interval set for SDLC time-out if the host does not poll this PU with a valid SDLC frame (default is 29).

Note: Unlike the other time-out parameters, SNA uses this time-out value along with the RETRY value.

See "SDLC Device Time-out Considerations" on page 2-11.

UNLOAD
This operand determines if $NETx is reloaded after unloading.

YES	Unload $NETx on attention **SNADACT/PUDACT**, if the host sends **DISCONNECT**, or if a permanent SDLC error condition occurs.
NO (default)	$NETx deactivates and unloads, then reloads and reactivates, if you receive a **DISCONNECT** from the host.
SNADACT	$NETx deactivates and unloads, then reloads and reactivates, if you receive a **DISCONNECT** from the host or if a permanent SDLC error condition occurs.

See "PU Reactivation" on page 2-18 and "SDLC Error Return Codes" on page 2-21.

Note: If you enter attention **SNADACT**, the $NETx and $SNA programs are unloaded. To unload $SNA, you must use the **SNADACT** operator command.

Warning:
For SNA Version 2.1, the $C $SNA or $C $NETx (where x is the number of the PU) commands are *not* allowed.

SNAPU Parameter Considerations

Prior to redefining the default values for SNAPU parameters, you should consider the following topics to determine which parameter values to specify when you define a PU.

Retry Considerations (RETRY)

The retry count specifies how many times SNA retries an unsuccessful I/O request for a PU before declaring an unrecoverable error situation.

SNA retries the following errors until successful:

- Busy after reset
- Busy.

SNA retries the following errors and, when the retry count is exceeded, considers them permanent errors:

- Data set errors
- Interface data checks
- Storage data checks
- Transmitted I-frames, negatively acknowledged
- Nonproductive receive time-out.

SNA never retries the following errors, which are immediately considered permanent errors:

- Device not attached
- Command reject
- Controller busy
- Delayed command reject
- Incorrect length
- DCB specification check
- Invalid storage address
- Protect check
- Clear-to-send time-out
- Data-set-ready time-out
- Data-terminal-ready time-out
- Controller end
- Program-controlled interrupt
- Attention with program-controlled interrupt.

Modem Considerations

Series/1 Synchronous Data Link Control requires a delay period of at least 8 to 16 milliseconds for turnaround time when using either modems or modem eliminators.

Hardware Support Considerations

The following parameters specify hardware support options.

CNCTYPE
Specifies the support to manage the connection of the communications lines. You can specify nonswitched (PP), auto-answer (AA), manual-answer (MA), auto-call (AC), direct-call (DC), or manual call (MC). The system assumes, as a default, that no special processing is required (nonswitched).

ENCODE
Specifies the bit transmission mode as "nonreturn to zero" (the system default) or "nonreturn to zero inverted." For successful results, both ends of the line must select the same bit transmission mode. For modems not supplying clocking, the SDLC adapter supplies internal clocking at 1200 bits per second and transmission mode of "non-return to zero inverted," regardless of your setting of the bit transmission mode indicator.

NORING
Since some modems provide ring interrupts and some modems do not, you have the option of selecting the ring support available on this system. The system assumes, as a default, that the modem presents a ring indication to the device when receiving an incoming call. The specification of the ring capability should agree with the specification of the NORING jumper on the SDLC attachment. The ring interrupt specification is meaningful only for switched lines.

PAD
Provides for inclusion of two synchronization characters on the initial transmission. The modem and communication lines being used may require this padding option. The system assumes as a default that pad characters are not included.

RATE
Sets the rate of the modem to full (the system default) or half speed. For successful operation of the line, the modem at each end of the line must be operating at the same rate.

See "Checklist" on page H-4 and "SDLC Specific Items" on page H-6 for more information.

SDLC Device Time-out Considerations

The SNAPU parameters enable you to select five hardware time-out values:

TOCTS
A clear-to-send time-out occurs if clear-to-send from the modem does not return within the prescribed time and the Series/1 attempts to send data.

TODSR
A data-set-ready time-out occurs if the SDLC ENABLE command does not complete within the prescribed time. Switched line operations may cause unexpected results if SNA uses this time-out. The ENABLE command requests that data-set-ready be brought up on the link. SNA issues the ENABLE command at network activation time but it does not complete until a call-in or call-out procedure successfully places the modem in data ready state. For nonswitched lines, the modem should always be in the data-ready state. During switch line initialization, the data set ready time-out is ignored and an infinite time-out is used while the SDLC support is waiting for the physical circuit to be established (auto answer with no ring, manual call, or manual answer).

TODTR
A data-terminal-ready time-out occurs if the SDLC DISABLE command does not complete within the prescribed time. DISABLE commands request that data-terminal-ready be dropped on the link.

TOHLA

A hold-line-active time-out provides a time period for the adapter to prepare for the next transmission operation in a chained transmission sequence. This is not an error time-out but defines how long the line should be held active after a transmit operation.

When using the #2080 feature attachment, it is recommended that the TOHLA time-out be a nonzero value.

TONPR

A nonproductive-receive time-out occurs if no valid SDLC frame was received from the host for this PU within the prescribed time. This time-out is a nonpermanent error until the RETRY count is exhausted; then SNA logs a permanent error. As an illustration, assume the default values for TONPR (29) and RETRY (3). The period of link inactivity before SNA logs a permanent error is:

```
TONPR + (TONPR x RETRY) = (29 x .106) + ((29 x .106) x 3),
which is approximately 12.3 seconds.
```

The SDLC time-outs are 1-byte values specified as quantities of timer units. Each timer unit is equivalent to 106 milliseconds. If you do not specify a value for each time-out, the system provides a default value of 29 timer units. If you specify a 0 value, the system selects an infinite time-out period. The maximum value permitted for each time-out specification is 255, which is approximately 27 seconds. The practical maximum in most cases is much less, determined by the characteristics of your network and links.

You must specify an absolute decimal number.

Threshold Value Considerations (THRESH)

The value specified for THRESH indicates the minimum number of buffers that must be available in the free buffer pool before the Series/1 can accept messages from the Network Control Program. This THRESH value prevents SDLC from depleting the free queue of buffers due to NCP send operations. The buffers remaining in the free buffer pool are available for other send operations. When the number of buffers in the pool is greater than the THRESH value, the Series/1 allows the Network Control Program to send messages. When the number of buffers in the pool is less than or equal to the THRESH value, the Series/1 does not allow the Network Control Program to send messages for this PU until sufficient buffers are available in the pool.

Buffers return to the free buffer pool when you issue NETGET instructions to obtain messages or when the Network Control Program acknowledges receipt of messages sent from the Series/1.

The decision to make THRESH large or small depends on:

- The number of frames per transmission.
- The number of messages expected per transmission from the Network Control Program.

A small THRESH value, that is, a value less than the maximum number of messages that can be sent by the NCP, allows the NCP to continue sending messages even though there are fewer buffers available than DCBs available to receive messages.

This does not cause problems if the NCP does not send more messages than the Series/1 has buffers in the free buffer pool. However, if the NCP sends more messages than the Series/1 has buffers, the excess messages are lost:

1. If more than one message is lost, the message carrying the Poll/Final flag is also lost. This results in the next transmission opportunity being lost.
2. The NCP times out and resends the Poll/Final flag.
3. The Series/1 responds with an acknowledgment indicating which messages were lost.
4. The NCP must retransmit the lost messages.

A larger THRESH value decreases the chances of losing messages, as described above. In fact, a THRESH value equal to the maximum number of messages that the NCP will send in one transmission ensures that no messages will ever be lost because of too few buffers. However, this value may not be the optimum value. If the Series/1 has enough buffers available to receive messages but not enough buffers available to satisfy the THRESH value, the NCP cannot send any messages until sufficient buffers are available in the free buffer pool to satisfy the THRESH value.

Figure 2-1 provides the advantages and disadvantages of specifying low or high THRESH values.

THRESH Value	Advantages	Disadvantages
Low THRESH	In light traffic, a few messages may still be received even if the buffer count is low.	If too many frames are received, only a few can be acknowledged. The Poll frame is lost. The NCP times out, repolls with a Receive Ready, and responds to the negative acknowledgment with a retransmission of all lost frames.
High THRESH	**Receive Not Ready** detects a low buffer count early and reports it to the NCP. Receive-Ready and Receive-Not-Ready frames swap until a low buffer count is resolved. There are no lost frames and no time-outs.	All inbound messages halt, even though some buffers may still be available to receive some messages.

Figure 2-1. THRESH Value Specification

Number of Frames per Transmission

To send or receive an SDLC frame requires one DCB and one buffer per frame. Use the following SNAPU parameters to make these selections:

DCBNO To select the number of DCBs.

BUFNO To select the number of buffers.

Consider the following items when you select the **maximum** DCBNO value:

- The rate of data flow in the link and the amount of data traffic to be serviced by the link.
- The frequency of transmission errors as dictated by the quality of the link.

Figure 2-2 summarizes the advantages and disadvantages of specifying few or many DCBs.

DCBNO Value	Storage	Data Flow	Link Errors
Few DCBs	Save 18 bytes per unspecified extra DCB. Save BUFSIZ bytes per buffer not needed in receive operations.	Frequent line turnaround. Poor rate of throughput in heavy traffic. Faster servicing of swapping Receive-Ready frames in light traffic.	Faster error recovery; since error on link causes minimum loss of frames, not many frames need be retransmitted.
Many DCBs	Each DCB uses 18 bytes. Larger BUFNO is required, using BUFSIZ bytes per buffer in receive operations.	Many frames transmitted before line turnaround. Good rate of throughput in heavy traffic. Slower servicing of swapping Receive Ready frames in light traffic since effort is required in setting of DCBs and buffers for receives that are not needed.	Slower error recovery: since error on link may cause loss of most of frames transmitted (error is not made known until end of transmission), retransmission of bad frames and all subsequent frames is required.

Figure 2-2. DCBNO Value Specification

SNA/SDLC Buffer Considerations

Use the BUFSIZ parameter to specify the physical size of the SDLC buffer (request unit). The size specified is the size of the largest acceptable request unit per frame. This value should be 9 less (6 bytes for the transmission header and 3 bytes for the request header) than the value specified for MAXDATA for the PU in the NCP definitions. BUFSIZ can be greater than MAXDATA−9, but it must not be less. That is, you can make BUFSIZ equal to MAXDATA−3, but you cannot make BUFSIZ equal to MAXDATA−13.

To make sure you have the maximum number of available buffers that may be required during inbound/outbound message processing, you can specify selected SNAPU and SNALU parameters so that the total number of buffers allocated (BUFNO + SENDBUF + RECVBUF) is greater than or equal to the results of the following formula:

```
(THRESH + 1) + (the maximum number of buffers allowed
for concurrent send operations) + (the number of receive buffers
required for inbound messages per session)
```

In the formula, the following apply:

1. "the maximum number of buffers allowed for concurrent send operations" (if buffer pooling) is the value you specify for SBUFNO. If SBUFNO=0, SNA uses the sum of the values specified on SENDBUF for all LUs defined. The maximum number of concurrent send buffers (if session buffering) is the value you specify on SENDBUF for a specific LU.
2. "the number of receive buffers required for inbound messages per session is the value resulting from the following formula:

   ```
   (INT((size of the largest inbound message)/(BUFSIZ)) * (2 * m−n)) per session
   ```

   where:

   INT means the integer result of the formula

   BUFSIZ is the RU size defined in the BIND command (and is the same as MAXDATA−9)

   m is the pacing group size

   n is the number of messages in the pacing group that has the pacing indicator (PI) set on.

   MAXDATA, the pacing group (m) and the pacing indicator (n) are defined in the NCP.

The formula takes into account when the host is segmenting messages to the Series/1.

The total number of buffers required can be minimized by specifying a smaller pacing group size and by setting on the PI in the last message of the pacing group.

The formula represents a maximum worst-case condition in which the following apply:

- Every session is utilizing the maximum number of send and receive buffers.
- All inbound messages are segmented using the maximum number of buffers per message.
- The Series/1 application is not issuing NETGETs.

If you limit the number of concurrent session buffer requirements, specify inbound/outbound pacing, and, for duplex operations, issue your NETGET task with a more privileged priority than your send task, you will probably require fewer buffers.

You should also specify your SNAPU and SNALU parameter relationships, as follows:

```
BUFNO + SENDBUF + RECVBUF > DCBNO > THRESH
```

The number of buffers allocated should be equal to or greater than the maximum number of consecutive frames specified (DCBNO). Otherwise, sufficient buffers may not be available to perform message exchange operations. Usually, you must specify BUFNO + SENDBUF +RECVBUF to be much greater than THRESH for throughput and performance.

The number of buffers you need depends on your operating environment. Each environment is different. You should evaluate your resources and objectives when defining your buffers.

For more information about buffer considerations, see Appendix I, "SNA Buffer Management."

Frame Count Considerations

Use the DCBNO parameter to specify the maximum number of frames (DCBs) that can be sent or received in a single I/O operation. If you do not specify a value, the system assumes a default of 2, the minimum number of frames required for successful system operation.

In general, the greater the message traffic, the higher the frame count:

- If there is high message traffic and you know there are few transmission errors, you can specify a high frame count because there will be less line turnaround.
- If there is high message traffic and you know there are many transmission errors, you should specify a low frame count, which causes fewer frames to be retried because of errors.
- If there is low message traffic and you specify a high frame count, system performance can be impacted.

While defining your network, you also specify the minimum number of buffers required for Receive-Ready state (THRESH). The value specified for THRESH must be less than the value specified for DCBNO. (The THRESH default is one.)

Buffer Pooling

In buffer pool management (BUFPOOL=YES), SNA and SDLC share a buffer pool that SNA allocates when starting a PU. During a session, you obtain buffers by issuing the send and receive operations. The system waits until a buffer is available and then completes the send operation. The maximum number of outstanding send buffers is the value you specify for SBUFNO. If SBUFNO=0, SNA uses the sum of the values specified on SENDBUF for all LUs. SBUFNO does not define more buffers (only the limit), whereas SENDBUF creates more buffers.

You can use send buffers for receive operations. All buffers defined are available for receive operations, but the number of buffers available for send operations is limited by SBUFNO or the sum of the SENDBUFs.

If no buffer is available for a receive operation, SDLC support sends Receive Not Ready (RNR). The system waits, and when a buffer becomes available SDLC support sends the message "Receive Ready" (RR). The system can then complete the receive operation.

See Appendix I, "SNA Buffer Management" for more information.

Session Buffering

In session buffer management (BUFPOOL=NO), the system allocates send and receive buffers per session when starting a PU. These buffers are available only for that specific session. If no buffer is available for a send operation, the session waits for one. When no buffer is available for a receive operation, SNA support ends the session.

Note: You must specify at least two send buffers and two receive buffers for each LU when using session buffering. Buffers are NOT shared between LUs.

See Appendix I, "SNA Buffer Management" for more information.

Segmented Message Buffer Considerations

When the Series/1 receives segmented messages, the number of segments in a message must not exceed the number of available buffers. SNA cannot free any buffers that contain message segments until receipt of the last segment of the message. If the number of segments exceeds the number of available buffers, SNA waits for an available buffer. The only way you can recover is to deactivate the network using the SNADACT operator command or by deactivating the PU at the host using the FORCE option.

See Appendix I, "SNA Buffer Management" for more information.

Switched Line Considerations

When you define a PU on a switched line, the NCP usually defines a unique ID (called the SDLC station identifier) for that PU within the network. When the connection is made, the NCP requests the station ID of the contacted PU and verifies it with the defined value for that PU in the NCP definitions. This allows the NCP to verify that the contacted PU is allowed to participate in the network. The STAXID parameter identifies the station ID for the PU, which must match the user-defined part of the station ID in the NCP. The PU block number or the machine type identifier for the Series/1 is X'021'.

PU Reactivation

If your Series/1 is unattended, you can eliminate the need for operator commands (PUACT) or EDL instructions (NETPACT) by defining the SNAPU UNLOAD parameter as UNLOAD=SNADACT. The value you specify for UNLOAD determines whether a PU reactivates. A PU deactivates because:

- The operator enters a PUDACT or SNADACT command.
- The system detects a permanent hardware error on the SDLC link.
- The host sends an SDLC DISCONNECT command.

Take action depending on the UNLOAD parameter as follows:

Reason for deactivation	UNLOAD=YES	UNLOAD=NO	UNLOAD=SNADACT
PUDACT/ SNADACT command	PU deactivates and unloads	PU deactivates and unloads	PU deactivates and unloads
SDLC hardware error	PU deactivates and unloads	PU deactivates and unloads	PU deactivates and unloads, reloads and reactivates
SDLC DISCONNECT command	PU deactivates and unloads	PU deactivates and unloads, reloads and reactivates	PU deactivates and unloads, reloads and reactivates

Concurrent Requests

At any one time, many SNA instruction requests can exist for a PU. Each request requires servicing of a PU stack. You specify the number of stacks with the STKNUM operand. The amount of storage required depends on the value you specify. If you decrease the value, SNA can issue more requests than there are stacks, forcing some requests to wait until previous requests finish and the PU releases the stack. This decreases performance.

Each LU used for a duplex session needs two stacks defined (one for sending data and one for receiving data). Each LU used for a half-duplex session needs one stack. After computing the total number of stacks needed, add one additional stack for termination of sessions. The actual number of stacks needed is usually much less. Not all LUs are in session simultaneously, and an LU rarely has more than one operation outstanding at a time, unless it is duplex.

The minimum number of stacks should be one more than the maximum number of concurrent sessions. The maximum number of concurrent sessions cannot be greater than the number of defined LUs. You can define more stacks for performance reasons and for duplex LUs.

SNALU Configuration Statement

The SNALU configuration statement generates an LU control block. You need an SNALU statement for each LU defined for a PU.

Prior to redefining the default values of this control block, you should consider the number of LUs to be specified for the PU interfacing with the network. This is discussed in "SNALU Parameter Considerations" on page 2-20.

The following are the syntax and parameter descriptions for the SNALU configuration statement:

```
label      SNALU     LU=,CTE=,END=,HOLDBND=,                          C
                     MSGPRIO=,RECVBUF=,SENDBUF=

Required: LU
Defaults: CTE=value of SENDBUF or CTE=2 if SENDBUF=0,
          END=NO,HOLDBND=NO,MSGPRIO=255,RECVBUF=0,SENDBUF=0
```

LU (required)
The LU number for this PU in the Series/1 to be used for the session when SNA issues NETINIT. This required parameter must be a decimal value (1 to 32), which must match the LOCADDR statement on the VTAM/NCP generation.

CTE
Specifies the maximum number (1 to the maximum available storage) of correlation table entries (CTEs) to allocate for this LU. If you do not specify a value, the value is the number of send buffers (SENDBUF=) allocated for this session. If you do not specify a value and SENDBUF=0, CTE defaults to 2. See "Correlation Tables" on page 13-7 for detailed information on correlation table entries.

END
Specifies whether it is the last LU for the PU.

YES Means this is the last LU for the PU.

NO (default) Means this is not the last LU for the PU.

HOLDBND
Specifies whether SNA should reject or hold a BIND received before a NETINIT is issued for the LU.

YES SNA holds the BIND and processes it upon receiving a NETINIT request with ACQUIRE=NO for the LU. If you issue NETINIT ACQUIRE=YES, SNA rejects the BIND received and sends INITSELF to request a new BIND.

NO (default) If a NETINIT has not been issued, SNA rejects the BIND and returns sense code 0801 (resource not available).

MSGPRIO
Specifies the priority of outbound messages for this LU. You can specify which message SDLC should service first if a bottleneck occurs. You must specify a value from 1 to 255. The default is MSGPRIO=255. If message priority is not important for a specific LU, then use the default value of 255 so as not to degrade overall performance.

RECVBUF
A decimal value that specifies the number of buffers to allocate for this particular LU at PU generation time. If you require session buffering (BUFPOOL=NO on the SNAPU statement), RECVBUF also represents the maximum number of outstanding inbound messages allowed for the specific LU. In this case, outstanding inbound messages refer to those for which your application has not issued NETGETS, but which SNA received anyway. If the number of outstanding inbound messages increases beyond the maximum allowed, the session ends. If you specify session buffering, RECVBUF must be at least 2. If you do not specify a value, RECVBUF defaults to zero. For more information, see "SNA/SDLC Buffer Considerations" on page 2-15.

SENDBUF
A decimal value that specifies the number of buffers to allocate for the LU at PU activation time. If you require session buffering (BUFPOOL=NO on the SNAPU statement), SENDBUF also represents the maximum number of outstanding outbound messages allowed for this specific LU. In this case, outstanding messages are those waiting to be transmitted by SDLC. If you specify session buffering, SENDBUF must be at least 2. If you do not specify a value, SENDBUF defaults to zero. For more information, see "SNA/SDLC Buffer Considerations" on page 2-15.

SNALU Parameter Considerations

When defining each PU, you must specify the number of LUs in the system. At PU activation, the SSCP attempts to establish a session with each LU of the Series/1.

In general, the number of LUs defined to the Series/1 should be the same as the number of LUs defined to the host for this cluster controller:

- If you define more LUs for any PU to the Series/1 than to the host, the SSCP is not aware of the extra Series/1 LUs. (For example, if you define 15 LUs to the Series/1 and 10 LUs to the host, the SSCP attempts to establish only 10 sessions.) In this case, the storage associated with the unused LUs defined to the Series/1 is not utilized. (See "Storage Requirements for SNA with SDLC Support" for information on LU storage sizes.)
- If you define fewer LUs for any PU to the Series/1 than to the host, the SSCP attempts to establish more sessions than the Series/1 wants established. (For example, if you define 10 LUs to the Series/1 and 15 LUs to the host, the SSCP attempts to establish 15 sessions.) In this case, the Series/1 rejects the requests for the extra LUs.

When defining LUs for the PU, you must specify END=YES on the last LU. If END=YES is not specified on the last LU, unpredictable results may occur during program execution.

Buffer Allocation

The values you specify for SENDBUF and RECVBUF define the number of buffers allocated for the LU. If BUFPOOL=NO, SNA allocates the buffers for use by that LU only. If BUFPOOL=YES, SNA adds these buffers to the PU buffer pool for use by any LU in the network. If you specify SBUFNO as a non-zero value on the SNAPU statement, you must specify SENDBUF and RECVBUF as zero. Otherwise, the sum of SENDBUF for each of the LUs is the maximum number of outstanding send operations allowed for the PU when buffer pooling.

See Appendix I, "SNA Buffer Management"for more information.

Disposition of BIND Requests

If an LU receives a BIND request before you issue a NETINIT for the LU, the HOLDBND parameter determines the disposition of the BIND. If HOLDBND=NO, SNA rejects the BIND with a 0801 sense code (resource not available). If HOLDBND=YES, SNA holds the BIND request and does not send a response until you issue a NETINIT for the LU. Use HOLDBND only for host-initiated sessions.

The disadvantage of specifying HOLDBND=YES is that SNA can leave a large number of BIND requests outstanding in the network. You must specify the maximum number of outstanding BINDs in the VTAM START macro ITLIM parameter (refer to the *VTAM Installation and Resource Definition* manual). If the ITLIM limit is exceeded, VTAM stops servicing user requests.

Prioritizing Messages

SNA interfaces with SDLC by a SEND queue. Each PU has a separate queue that handles messages for that PU. SNA orders the queue by priority, with 0 the highest priority and 255 the lowest. SNA reserves 0 priority for PU messages. Assigning high priority to LUs with interactive sessions and low priorities to batch sessions allows SNA to send the interactive messages first if a bottleneck occurs in SDLC.

To maintain correct SNA architecture, SNA places a message from an LU after all other messages from that LU that are still on the send queue, even if the priority is higher.

Note: If message priority is not important for a specific LU, then use the default value of 255 so as not to degrade overall performance.

SDLC Error Return Codes

The following list is the SDLC device error return codes and the probable causes of the error.

Note: These return codes are not for users of shared SDLC support. Users of shared SDLC support should refer to the *Messages and Codes* manual for the error return codes

Return Code	Condition
-371	Error on read cycle steal status operation.
-360	You attempted an X.21 operation, but your system does not contain X.21 circuit switched network support.
-359	Device not jumpered correctly for connection type.
-358	Invalid connection type specified for SDLC device.
-354	Enable/disable failed.
-351	I/O reset failed.
1	Retry of enable/disable count exceeded.
2	Time-out on switched line.
3	Receive data set error, retry limit exceeded.

4	Send error retry count limit exceeded.
5	2080 or 2090 attachment card failed.
6	Disaster ISB, like DCB spec check.
7	OIO retry failed
8	Single frame retransmit limit exceeded.
9	X.21 connection attempt failed or DCE CLEAR received from network.
10	SNA/SDLC detected an error in the hardware configuration.
11	Non-productive receive time-out; retry limit exceeded.

See "X.21 Return Codes and Messages" on page 18-10 for an explanation of X.21 return codes.

Storage Requirements for SNA with SDLC Support

Figure 2-3 describes the storage requirements for EDX SNA Version 2.1 with SDLC support within a Series/1. It describes the calculations needed to estimate the storage requirements of your customized configuration. The PU control blocks for each PU can be in separate partitions, and $SNA need not be in the same partition with any PU. SNA allocates a 1-page program control data area with a label of **DATA** in partition number 1 when $SNA is loaded.

Description	# of bytes
Storage requirements for $SNA: SNA resident code with SDLC support	 37,748
Storage requirements for each PU ($NETx): Resident code to support the PU PU control blocks Stack area # of stacks (STKNUM) x 348 bytes DCB area # of DCBs (DCBNO) x 18 bytes LU area (# of LUs (SNALU x 350 bytes) + (8 bytes x # of CTEs (CTE))) Buffer area (# of buffers (BUFNO + SENDBUF + RECVBUF) x buffer size (BUFSIZ + 24)) Total storage for PU (round up to page boundary)	 4072 884
Storage requirements for each SNA application: $NETCMD (see note 1) $NETPACT (see note 2)	 2664 1190

Figure 2-3. Storage Requirements

Figure 2-4 shows a sample calculation, using the starter system needed to estimate the storage requirements of your customized configuration.

Notes:

1. Required if application issues NETINIT, NETPUT, NETGET, NETCTL, or NETTERM instructions.
2. Required if application issues NETPACT instruction.

Sample Storage Calculation

Storage requirements for $SNA:	
SNA resident code without X.21 support	37.75K bytes
Storage requirements for each PU ($NETx):	
Resident code to support the PU	4072 bytes
PU control blocks	884 bytes
Stack area (2 x 348 bytes)	696 bytes
DCB area (2 x 18 bytes)	36 bytes
LU area (1 x (350 bytes + (8 bytes x 2)))	366 bytes
Buffer area ((3 + 2 + 2) x (256 + 24 bytes))	1960 bytes
Total storage for PU (rounded to page boundary)	8.25K bytes
Storage requirements for each SNA application	
$NETCMD	2664 bytes
$NETPACT	1190 bytes

Figure 2-4. Sample Storage Calculation

SDLC Device Error Logging

The Series/1 SNA/SDLC support logs recoverable and unrecoverable errors in the system error log if error logging is active. SDLC logtype entries contain X'01' for recoverable errors and X'03' for unrecoverable errors. Unrecoverable error log entries are always displayed formatted. Unrecoverable error log entries contain 104 bytes of log information, and recoverable error log entries contain up to 256 bytes of error log information. Only unrecoverable error log entries are formatted by the PL or LL commands of $DISKUT2.

To start an error log, you must use the EDX $LOG utility. In EDX Version 5 and subsequent versions, $LOG is always active unless you explicitly end it. Refer to the *Problem Determination Guide* or *Operator Commands and Utilities Reference* for more information on $LOG.

The error log record contains information on the number of retries for the SDLC device, the original I/O condition codes, interrupt condition codes, cycle steal status words, immediate device control blocks, and device control blocks. To print a copy of the log, use the PL command of $DISKUT2. To view the log on your terminal, use the LL command of $DISKUT2. The error log supplies additional diagnostic information that assists IBM if you need to place a service call.

Log Record Format for Recoverable SDLC Device Errors

The field descriptions for an SDLC recoverable log entry error are as follows:

COMMON LOG AREA

Offset	Description
X'00'	Logtype—X'01' – hardware recoverable
X'01'	Interrupt condition code
X'02'	Interrupt status byte
X'03'	OIO condition code
X'04'	Device address
X'06'	Device type error
X'07'	Reserved
X'08'	Reserved
X'0A' – X'19'	SNA internal use only
X'1A'	DDB address

DDB LOG AREA

Offset	Description
X'1C'	Flags status of last retry
X'1E'	Device type ID
X'20'	Original IDCB address
X'22'	Retry count
X'23'	Original OIO condition code
X'24'	Original interrupt condition code
X'25'	Original interrupt status byte
X'26'	Cycle steal status word count (maximum of 10)
X'27'	Bytes of status information (maximum of 6)

DDB SDLCOM EXTENSION LOG AREA

Offset	Description
X'28' – X'2D'	SNA internal use only
X'2E'	Number of SDLC buffers
X'30' – X'49'	SNA internal use only
X'4A'	Residual address
X'4C'	Residual byte count

X'4E'	Start cycle steal status flags	
	X'8000'	
		0 = no overrun in receive
		1 = overrun in receive
	X'4000'	
		0 = no abort in receive
		1 = abort in receive
	X'2000'	
		0 = no long frame error
		1 = long frame error
	X'1000'	
		0 = no block check error in receive
		1 = block check error occurred within SDLC
	X'0800'	
		0 = no time-out occurred
		1 = time-out occurred for data set ready, clear to send, or disable data terminal ready
	X'0400'	
		0 = no idle detect received
		1 = idle detect received
	X'0200'	
		0 = no nonproductive received
		1 = nonproductive received
	X'0100'	
		0 = no data set ready time-out
		1 = data set ready time-out
	X'0080'	reserved
	X'0040'	reserved
	X'0020'	reserved
	X'0010'	reserved
	X'0008'	reserved
	X'0004'	
		0 = no business machine clock
		1 = business machine clock

		X'0002'
		0 = answer tone jumpered off
		1 = answer tone jumpered on
		X'0001' reserved
	X'50'	Start cycle steal status flags
		X'8000'
		0 = not data terminal ready
		1 = data terminal ready
		X'4000'
		0 = not data set ready
		1 = data set ready
		X'2000'
		0 = not request to send
		1 = request to send
		X'1000'
		0 = not clear to send
		1 = clear to send
		X'0800'
		0 = not noring indicator
		1 = noring indicator
		X'0400'
		0 = not half rate selected
		1 = half rate selected
		X'0200'
		0 = not transmit mode latch
		1 = transmit mode latch
		X'0100' reserved
	X'51'	Secondary station ID
	X'52'	Disaster buffer return code
	X'53'	Reserved

DLD CONTROL BLOCK LOG AREA

Offset	Description
X'54'	SNA internal use only
X'56'	Flag bits and switches
	X'8000'
	0 = leased connection
	1 = switched connection
	X'4000'
	0 = host should dial the Series/1 (answer)
	1 = Series/1 should dial the host (call)
	X'2000'
	0 = manual answer
	1 = auto answer
	X'1000'
	0 = normal processing
	1 = a permanent error occurred within SDLC
	X'0800'
	0 = normal processing
	1 = a permanent error was detected but was not yet processed
	X'0400'
	0 = normal processing
	1 = SDLC connection closed by SNA
	X'0300' Possible values of the mask:
	B'00' Reactivate the PU if the host sends the SDLC DISCONNECT command
	B'01' Reactivate the PU if the host sends the SDLC DISCONNECT command or a permanent error occurs in the network
	B'10' Do not reactivate the PU
	B'11' Reserved
	X'0080'
	0 = use full modem rate
	1 = use half modem rate

X'0040'

0 = use NRZ data encoding

1 = use NRZI data encoding

X'0020'

0 = reserved

1 = reserved

X'0010'

0 = ignore all attention interrupts (NORING=YES)

1 = enable the adapter if SDLC receives an attention interrupt (NORING=NO)

X'0008'

0 = do not insert pad characters in the data

1 = insert pad characters in the data

X'0004'

0 = this is a secondary SDLC station

1 = reserved

X'0002'

0 = no elements on SDLC read queue for SNA

1 = SDLC read queue contains data for SNA

X'0001'

0 = NORING jumpered off

1 = NORING jumpered on

X'58'	Return code
X'5A'	SNA internal use only
X'5C'	Address of first buffer in free pool
X'5E'	Address of last buffer in free pool
X'60'	Address of first buffer on READ queue
X'62'	Address of last buffer on READ queue
X'64'	Second flag byte

X'8000'

0 = no nonproductive receive time-out before first poll

1 = nonproductive receive time-out before first poll

X'65'	SNA internal use only
X'66'	Maximum I-field this response
X'68'–X'6F'	SNA internal use only
X'70'	Threshold value
X'71'	Disable data terminal ready time-out
X'72'	Data set ready time-out
X'73'	Request on line response time-out
X'74'	Idle detect time-out
X'75'	Nonproductive receive time-out
X'76'	Clear to send time-out
X'77'	Hold line active time-out
X'78'	Current DCB being serviced
X'7A'	Command field of IDCB
X'7B'	Device field of IDCB
X'7C'	Address of first DCB
X'7E'	Start cycle steal status command
X'7F'	Device address in RCSS IDCB
X'80'	Address of RCSS DCB
X'82'	Control flags
X'84'	Timer one
X'85'	Timer two
X'86'	Residual count
X'88'	Residual status flags
X'8A'	Residual address pointer
X'8C'	Address of next DCB
X'8E'	Byte count of data area
X'90'	Data area address
X'92'	Start of normal DCBs

Device-Dependent Data Section of an SDLC Unrecoverable Error Log

The following illustrates the device-dependent section of an SDLC permanent error log.

Byte 0 Flag word 1 (reserved)

Byte 1 SDLC disaster return code. See positive return codes under "SDLC Error Return Codes" on page 2-21 for possible values of the field.

Byte 2 Flag word 2 (bit significant)

X'8000'

0 = leased connection

1 = switched connection

X'4000'

0 = host should dial the Series/1 (answer)

1 = Series/1 should dial the host (call)

X'2000'

0 = manual answer

1 = auto answer

X'1000'

0 = normal processing

1 = a permanent error occurred within SDLC

X'0800'

0 = normal processing

1 = a permanent error was detected but was not yet processed

X'0400'

0 = normal processing

1 = SDLC connection closed by SNA

X'0300' Possible values of the mask:

B'00' Reactivate the PU if the host sends the SDLC DISCONNECT command

B'01' Reactivate the PU if the host sends the SDLC DISCONNECT command or a permanent error occurs in the network

B'10' Do not reactivate the PU

B'11' Reserved

X'0080'

0 = use full modem rate

1 = use half modem rate

X'0040'

0 = use NRZ data encoding

1 = use NRZI data encoding

X'0020'

0 = reserved

1 = reserved

X'0010'

0 = ignore all attention interrupts (NORING=YES)

1 = enable the adapter if SDLC receives an attention interrupt (NORING=NO)

X'0008'

0 = do not insert pad characters in the data

1 = insert pad characters in the data

X'0004'

0 = this is a secondary SDLC station

1 = reserved

X'0002'

0 = no elements on SDLC read queue for SNA

1 = SDLC read queue contains data for SNA

X'0001'

0 = NORING jumpered off

1 = NORING jumpered on

Byte 4 Flag word 3 (bit significant)

X'8000'

0 = SDLC in disconnected mode (SNRM not received)

1 = SDLC in normal-response mode (SNRM received, able to exchange data)

X'4000'

0 = Series/1 sends next SDLC frame

1 = host must send the next SDLC frame

X'2000'

0 = Series/1 SDLC is in receive-not-ready state (may not accept SDLC I-frames)

1 = Series/1 is in receive-ready state (may accept I-frames)

X'1000'

0 = host SDLC is in receive-not ready state (Series/1 may not send SDLC I-frames)

1 = host SDLC is in receive-ready state (Series/1 may send I-frames)

X'0800'

0 = reserved

1 = reserved

X'0400'

0 = reserved

1 = reserved

X'0200'

0 = normal processing

1 = illegal I-field (I-field present on an SDLC command that cannot have one)

X'0100'

0 = normal processing

1 = command reject state

X'0080'

0 = reserved

1 = reserved

X'0040'

0 = do not poll

1 = polling active

X'0020'

0 = normal processing

1 = UA expected or just sent

X'0010'

0 = normal processing

1 = TEST command received, response due

X'0008'

0 = normal processing

1 = XID command received, response due

X'0004'

0 = normal processing

1 = SNA in process of stopping SDLC

X'0002'

0 = reserved

1 = reserved

X'0001'

0 = validate number received (NR) count

1 = NR count in frame checked

Byte 6 Flag word 4 (bit significant)

X'8000'

0 = normal processing

1 = error encountered during a receive operation

X'4000'

0 = normal processing

1 = error encountered during a send operation

X'2000'

0 = normal processing

1 = error encountered during an enable operation

X'1000'

0 = normal processing

1 = error encountered during a disable operation

X'0800'

0 = normal processing

1 = DISCONNECT received from host

X'0400'

0 = not X.21 auto call

1 = X.21 auto call requested (CNCTYPE=AC)

X'0200'

0 = not X.21 direct call

1 = X.21 direct call requested (CNCTYPE=DC)

X'0100'

0 = SDLC cannot send or receive SDLC frames

1 = SDLC can send and receive SDLC frames

X'0080'

0 = not using X.21 switched connection

1 = X.21 switched connection requested

X'0040'

0 = X.21 switched not active

1 = X.21 switched connection made

X'0020'

0 = normal processing

1 = error occurred during X.21 switched

X'0010'

0 = no X.21 connection error

1 = error occurred while trying to establish an X.21 switched connection

X'0008'

0 = normal processing

1 = error occurred preparing the SDLC device for use

Byte 8 X'03'

Byte 9 Reserved

Byte 10 Number of successfully transmitted frames

Byte 12 Number of successfully received frames

Byte 14 Number of successfully transmitted frames with the poll/final indicator on in the frame

Byte 16 Number of test frames sent

Byte 18 Number of receive-not-ready frames sent

Byte 20 Number of times an overflow error occurred while receiving a frame.

Byte 22 Number of times an abort error occurred while receiving a frame

Byte 24 Number of long frames received

Byte 26 Number of times a frame was lost because of a block check error

For additional information on error logging and how to use it, refer to the *Problem Determination Guide* and the *Operator Commands and Utilities Reference*

See "X.21 Error and Call Progress Signal Logging" on page 18-6 for information on error logging when using the X.21 circuit switched network support.

Chapter 3. Installing and Defining the SNA Network for Use with Shared SDLC

This chapter tells you how to install SNA for use with shared SDLC on your system, how to prepare your EDX system for SNA with shared SDLC, and how to define your SNA network for use with shared SDLC. The chapter contains the shared SDLC return codes, and storage requirements. Refer to the EDX SNA Program Directory for a more detailed discussion and step-by-step instructions for installing SNA and defining SNA to EDX.

Refer to the EDX Version 6 program directory for information regarding shared SDLC installation.

> **Warning:**
> If you do not copy the object modules from the diskette,
> you must keep the diskette to apply APAR/PTF module replacements,
> which require a relink of the SNA main program.
> DO NOT DISCARD THE DISKETTE CONTAINING THESE MODULES.

Note: Unlike Version 1.2 of SNA, the EDX SNA object modules (except the ones copied automatically) are **not** required to define and configure your SNA system.

If you leave the object modules on diskette, you save disk space. Only the modules needed to assemble and build your SNA definitions and applications are automatically copied to your system.

Installing SNA

You can use SNA Version 2.1 with shared SDLC support only with an EDX Version 6 system. If you are installing SNA Version 2.1 with shared SDLC on an EDX Version 6 system, you must run $INSTAL to install the required SNA Version 2.1 modules. $INSTAL is an installation utility available with the EDX Version 6 product. $INSTAL maintains a history file of the current level of software on a system. $INSTAL can assist you in determining the version, modification level, and the APAR fixes on each system.

If you do not use $INSTAL to edit the default installation specifications, the EDX SNA object modules are left on diskette and the $S1ASM macros are not copied. The only object modules automatically copied are the ones needed to assemble and build your SNA definitions and applications.

To change the installation specifications, you must use $INSTAL to edit the $$INXX9 control data set. By editing the $$INXX9 data set, you can specify that the EDX SNA object modules and/or the $S1ASM macros should be copied to disk.

To specify copying all SNA object modules to disk, use $INSTAL to edit $$INXX9 on volume XX9001. When prompted, change the $COPYUT1 control file name from "XX9001N" to "XX9001" on volume XX9001. To specify copying the $S1ASM SNA macros, add the $COPYUT1 control file names "XX9002" on volume EDX002 and "XX9003" on volume EDX002.

$INSTAL copies the prebuilt $SNAEX program onto EDX002. The $SNAEX program contains the SNA Version 2.1 modules with the extended DLM support, which enables you to use the shared SDLC support. You must rename the $SNAEX program to $SNA using $DISKUT1.

Refer to the SNA program directory for more detailed installation procedures, and an example of changing the default installation specifications.

Installing EDX Version 6 Required Support

To use Version 2.1 of SNA with shared SDLC support, you must install the following EDX Version 6 support. This support is available on the XS6007 EDX Version 6 diskette.

- Data link router support—DLCROUTE
- Communication Common Services support
- Shared SDLC support.

Defining SNA to EDX

To use SNA with shared SDLC support in your EDX system, you must define each SDLC attachment card you require:

- If you use hardware station address recognition or the secondary station address jumpered on the SDLC attachment card, then you must define one EXIO device for each PU (up to a maximum of four) during the system generation process. Each PU requires a separate SDLC attachment card. Only one station address is assigned to each attachment card when using the secondary station address jumpered on the card.
- If you use the secondary station address as specified by the NDU SDLC station configuration record, then more than one PU can share an SDLC attachment card by using the shared link support in shared SDLC. If you are not using shared link support in shared SDLC, then each PU requires its own SDLC attachment card. Each SDLC attachment card requires an EXIO device defined during system generation. See "SNA Sharing a Link with APPC or Another SNA PU" on page 3-19 for more information.

Do the following for an EDX SNA system generation:

- Define the SNA EDL instructions to the EDX overlay assembler ($EDXASM) by adding the EXTLIB statement for $EDXLSNA to $EDXL if you use the overlay assembler to define your network or assemble your application programs.
- If you use $S1ASM instead of $EDXASM to define your network and assemble your applications, you must copy the macros from diskette onto your system. Otherwise, you do not need to include the macros in your system.
- Edit the $EDXDEF data set on volume ASMLIB to match the hardware required by your SNA and shared SDLC support. You should specify an EXIODEV statement for each SDLC device attachment card on your Series/1.
  - If you are not using the shared link support in shared SDLC, then each active PU requires a separate SDLC attachment card.
  - Specify the MECBLST parameter on the SYSPARMS statement. SNA requires that the value for the MECBLST parameter be at least 1.

- Each EXIODEV statement for an SDLC attachment must specify RSB=4 and MAXDCB=16.

  Note: The value specified for the MAXOUT parameter of GROUP, LINE, or PU macros for the Network Control Program (NCP) generation must be 7, as in the following example:

```
EXIODEV    ADDRESS=0A,RSB=4,MAXDCB=16

PU         MAXOUT=7,...          (NCP macro)
```

- Edit the EDX $LNKCNTL data set to include support for SNA and shared SDLC in your system. Include the following object modules in your supervisor:
  - **SWAITM**—for wait-on-multiple-event support
  - **IOSEXIO**—for EXIO device support
  - **EDXTIMER** or **EDXTIMR2**—for timer support
  - **FULLMSG**—for full message data set support
  - **DLCROUTE**—for data link router support.
- Complete the system generation as described in the *IBM Series/1 Event Driven Executive Installation and System Generation Guide.*

Notes:

1. SNA requires timer support in the supervisor. SNA does not require a timer card.
2. If timer support, full message support, wait-on-multiple-event support, or data link router support are not included in the EDX supervisor, then SNA will print an error message and unload.
3. The DLCROUTE include can be in any partition in a nonoverlay area.

Defining the SNA Network

You define your SNA network by defining PUs and LUs. In defining your network for use with shared SDLC, you must do the following:

1. Edit $SNADEFx
2. Edit LINKNETx
3. Run $JOBUTIL.

To define PUs, you use the SNAPU configuration statement to generate a PU control block; to define SDLC, you use the configuration records of the Network Definition Utility.

You can define one or multiple PUs. To define a PU, you must assemble and link $SNADEFx (*x* is the number of the PU), the Series/1 SNA configuration file, using BLDNETx.

Editing $SNADEFx

When you edit $SNADEFx, you do the following:

- Edit the $SNADEFx data set. The $SNADEFx data set contains the definitions for the SNAPU and SNALU configuration statements. See "SNAPU Configuration Statement for SNA with Shared SDLC Support" on page 3-9 and "SNALU Configuration Statement" on page 3-17 for more details about editing the SNAPU and SNALU statements.
- If you use only one PU, place your network definitions in $SNADEF1. $SNADEF1 is shipped with the product and contains default values for the PU and LU.

  Note: The default values specified in $SNADEF1 compose the default SNA network configuration defined in the SNA Program Directory. The default network is for SNA with SDLC support. If you want to use SNA with shared SDLC support, you must edit the $SNADEF1 data set and remove all SDLC dependent information. The SDLC dependent information for shared SDLC is defined in configuration records built using the Network Definition Utility.
- If you use multiple PUs (up to a maximum of four), place each PU definition in $SNADEFx, where *x* is the number of the PU. Use $SNADEF2 for PU #2, $SNADEF3 for PU #3, and so on.

Editing LINKNETx for Shared SDLC

Before building the $NETx program, you must edit LINKNETx (x is the number of the PU you are defining) to include the module CDCBRWTM. This module is required for the extended DLM support.

Note: If you do not include CDCBRWTM, the $EDXLINK will succeed, but the $NETx will receive a program check when loaded.

When editing LINKNETx, you can comment out modules that are not necessary for shared SDLC support. Comment out the following statements:

- $EDXEXIO
- CDCLOG
- CDETCB
- $DEVLOG

If you leave these modules included, then no error occurs. However, you can decrease the amount of storage $NETx requires by removing them.

Figure 3-1 on page 3-5 shows a listing of the $EDXLINK control cards for LINKNET1. The control cards for $NET1 are edited for use with shared SDLC.

```
  LIST OF DATA SET LINKNET1 ON EDX003
*****************************************************************
*                                                               *
*  NET  -  LINK EDIT STATEMENTS FOR SNA CONTROL BLOCKS ($NET1)* 
*                                                               *
*****************************************************************
*
AUTOCALL $AUTO,ASMLIB
INCLUDE $NETBLD1,EDX003
*NCLUDE $EDXEXIO,EDX003          NOT REQUIRED FOR SHARED SDLC
INCLUDE CDCBRWTM,EDX003          REQUIRED FOR SHARED SDLC
*NCLUDE CDCLOG,EDX003            NOT REQUIRED FOR SHARED SDLC
INCLUDE CDDMSG,EDX003
INCLUDE CDDQON,EDX003
INCLUDE CDDTOSNA,EDX003
INCLUDE CDDATA00,EDX003
*NCLUDE CDETCB,EDX003            NOT REQUIRED FOR SHARED SDLC
INCLUDE $SNAERR,EDX003
*NCLUDE $DEVLOG,ASMLIB           NOT REQUIRED FOR SHARED SDLC
INCLUDE $$RETURN,ASMLIB
INCLUDE $$SVC,ASMLIB
INCLUDE ASMOBJ,EDX002
LINK $NET1,EDX002 REPLACE END
```

Figure 3-1. $EDXLINK Control Cards for Link Editing $NET1 (Example)

Running $JOBUTIL

After editing $SNADEFx and LINKNETx, you must build the PU using the $JOBUTIL control cards in BLDNETx (x is the number of the PU you are defining). BLDNET1 contains the $JOBUTIL control cards needed to assemble the definitions, link them with other modules, and store the result in $NET1. $JOBUTIL creates programs named $NETx, where *x* is the number of the PU containing your PU definitions. SNA reserves the names $NET1 through $NET4, and you must not change them.

Defining Default $NETx (Example)

The $SNADEF1 data set is used to create a customized network PU ($NET1) that fits the user's needs. The sample $SNADEF1 provided on the installation diskette appears as follows. This is the definition of the network PU ($NET1) provided on the installation diskette. This sample network consists of one LU and one PU. To use shared SDLC support, you must edit this data set.

```
NET1     CSECT ,                 DEFINE THE NETWORK
         SNAPU RETRY=3,          NUMBER OF I/O RETRIES                X
               ADDRESS=0A,       DEVICE ADDRESS (HEXADECIMAL)         X
               CNCTYPE=PP,       CONNECTION TYPE                      X
               NORING=NO,        INCLUDE RING SUPPORT                 X
               RATE=FULL,        MODEM SPEED                          X
               ENCODE=NRZ,       MODEM ENCODING                       X
               PAD=NO,           SEND PAD CHAR ON INITIAL XMISSION    X
               UNLOAD=YES,       UNLOAD $SNA IF DISC/PERM ERROR       X
               TODTR=0,          DATA-TERMINAL-READY TIMEOUT          X
               TODSR=0,          DATA-SET-READY TIMEOUT               X
               TOCTS=0,          CLEAR-TO-SEND TIMEOUT                X
               TOHLA=0,          HOLD-LINE-ACTIVE TIMEOUT             X
               TONPR=0,          NON-PRODUCTIVE RECEIVE TIMEOUT       X
               FRTPOLL=NO,       TIMEOUT ON FIRST POLL                X
               THRESH=2,         MINIMUM BUFFS TO SEND RECEIVE-READY  X
               BUFSIZ=256,       SIZE OF LARGEST USER MESSAGE         X
               BUFNO=3,          NUMBER OF BUFFERS AT ACTIVATION      X
               DCBNO=2,          MAX NUM OF FRAMES TO SEND OR RECV    X
               STAXID=S00000,    ID TO SEND IN RESPONSE TO XID CMD    X
               NETNAME=NET1,     NETWORK NAME THIS CONNECTION         X
               BUFPOOL=YES,      BUFFER POOLING                       X
               STKNUM=2,         NUMBER OF STACKS FOR LUS             X
               SBUFNO=0          CONCURRENT SEND BUFFERS
         EJECT
         SNALU LU=1,             LOGICAL UNIT NUMBER                  X
               SENDBUF=2,        NUMBER OF SEND BUFFERS FOR LU        X
               RECVBUF=2,        NUMBER OF RECEIVE BUFFERS FOR LU     X
               CTE=2,            NUMBER OF CORRELATION TABLE ENTRIES  X
               HOLDBND=NO,       HOLD BIND UNTIL NETINIT ISSUED       X
               END=YES           LAST LOGICAL UNIT TO GENERATE
         END
```

To define and generate the above SNA network with the shared SDLC support, use $FSEDIT to edit the data set $SNADEF1 on EDX003 from the installation diskette and specify the following:

```
NET1     CSECT ,                 DEFINE THE NETWORK
         SNAPU CNCTYPE=EX,       CONNECTION TYPE                      X
               CNCNAME=NETSDLC1, NAME OF CONFIGURATION RECORD         X
               UNLOAD=YES,       UNLOAD $SNA IF DISC/PERM ERROR       X
               BUFSIZ=256,       SIZE OF LARGEST USER MESSAGE         X
               BUFNO=3,          NUMBER OF BUFFERS AT ACTIVATION      X
               STAXID=S00000,    ID TO SEND IN RESPONSE TO XID CMD    X
               NETNAME=NET1,     NETWORK NAME THIS CONNECTION         X
               BUFPOOL=YES,      BUFFER POOLING                       X
               STKNUM=2,         NUMBER OF STACKS FOR LUS             X
               SBUFNO=0          CONCURRENT SEND BUFFERS
         EJECT
         SNALU LU=1,             LOGICAL UNIT NUMBER                  X
               SENDBUF=2,        NUMBER OF SEND BUFFERS FOR LU        X
               RECVBUF=2,        NUMBER OF RECEIVE BUFFERS FOR LU     X
               CTE=2,            NUMBER OF CORRELATION TABLE ENTRIES  X
               HOLDBND=NO,       HOLD BIND UNTIL NETINIT ISSUED       X
               END=YES           LAST LOGICAL UNIT TO GENERATE
         END
```

The following is a listing of the $EDXLINK control cards for link-editing $NET1.

```
  LIST OF DATA SET LINKNET1 ON EDX003
*****************************************************************
*                                                               *
*  NET  -  LINK EDIT STATEMENTS FOR SNA CONTROL BLOCKS ($NET1)* 
*                                                               *
*****************************************************************
*
AUTOCALL $AUTO,ASMLIB
INCLUDE $NETBLD1,EDX003
INCLUDE $EDXEXIO,EDX003          NOT REQUIRED FOR SHARED SDLC
*NCLUDE CDCBRWTM,EDX003          REQUIRED FOR SHARED SDLC
INCLUDE CDCLOG,EDX003            NOT REQUIRED FOR SHARED SDLC
INCLUDE CDDMSG,EDX003
INCLUDE CDDQON,EDX003
INCLUDE CDDTØSNA,EDX003
INCLUDE CDDATA00,EDX003
INCLUDE CDETCB,EDX003            NOT REQUIRED FOR SHARED SDLC
INCLUDE $SNAERR,EDX003
INCLUDE $DEVLOG,ASMLIB           NOT REQUIRED FOR SHARED SDLC
INCLUDE $$RETURN,ASMLIB
INCLUDE $$SVC,ASMLIB
INCLUDE ASMOBJ,EDX002
LINK $NET1,EDX002 REPLACE END
```

To use shared SDLC, you must use $FSEDIT to edit LINKNET1 on EDX003 to specify the following:

```
  LIST OF DATA SET LINKNET1 ON EDX003
*****************************************************************
*                                                               *
*  NET  -  LINK EDIT STATEMENTS FOR SNA CONTROL BLOCKS ($NET1)* 
*                                                               *
*****************************************************************
*
AUTOCALL $AUTO,ASMLIB
INCLUDE $NETBLD1,EDX003
*NCLUDE $EDXEXIO,EDX003          NOT REQUIRED FOR SHARED SDLC
INCLUDE CDCBRWTM,EDX003          REQUIRED FOR SHARED SDLC
*NCLUDE CDCLOG,EDX003            NOT REQUIRED FOR SHARED SDLC
INCLUDE CDDMSG,EDX003
INCLUDE CDDQON,EDX003
INCLUDE CDDTOSNA,EDX003
INCLUDE CDDATA00,EDX003
*NCLUDE CDETCB,EDX003            NOT REQUIRED FOR SHARED SDLC
INCLUDE $SNAERR,EDX003
*NCLUDE $DEVLOG,ASMLIB           NOT REQUIRED FOR SHARED SDLC
INCLUDE $$RETURN,ASMLIB
INCLUDE $$SVC,ASMLIB
INCLUDE ASMOBJ,EDX002
LINK $NET1,EDX002 REPLACE END
```

To create the network PU program, you must load the utility program $JOBUTIL. Load the program as shown in the following example:

```
>$L $JOBUTIL BLDNET1,EDX003
LOADING $JOBUTIL     nnP, hh:mm:ss, LP= xxxx, PART= yy
```

Note: See "Utilities Examples" on page xviii for more information about the symbols used in the above examples.

The BLDNET1 data set contains the job cards to do the following:

- Assemble $SNADEF1.
- Convert the result into an executable program ($NET1) using $EDXLINK.

Repeat the procedure shown above for each network PU program you must create, assigning a new number from 1 to 4. For PU #1, use BLDNET1, LINKNET1, and $SNADEF1. For PU #2, use BLDNET2, LINKNET2, and $SNADEF2, and so on.

Defining the PU

To use SNA with shared SDLC support, you must define the SNA network (PUs and LUs) and the SDLC configuration records. For the SNA PUs, this means editing the SNAPU configuration statement. Since the Network Definition Utility is used to define the SDLC characteristics, the SNAPU must have all the SDLC-dependent information removed. The advantage to using the Network Definition Utility product is that you can dynamically change the SDLC characteristics by redefining the network configuration records. If you need to change your SDLC characteristics only, you use the Network Definition Utility and do not have to rebuild your $NETx program.

Defining the PU Using Shared SDLC

Use $FSEDIT to edit the SNAPU configuration statement. You must remove the following parameters, which are not valid when you use SNA with shared SDLC support:

- ADDRESS
- DCBNO
- ENCODE
- FRTPOLL
- NORING
- PAD
- RATE
- RETRY
- THRESH
- TOCTS
- TODSR
- TODTR
- TOHLA
- TONPR.

SNAPU Configuration Statement for SNA with Shared SDLC Support

The SNAPU configuration statement generates a PU control block. The following are the syntax and parameter descriptions for the SNAPU configuration statement for SNA with shared SDLC support. (Before you redefine the default values for SNAPU parameters, review "SNAPU Parameter Considerations" on page 3-11.)

```
label      SNAPU     BUFNO=,BUFPOOL=,BUFSIZ=,CNCNAME=,CNCTYPE=,        C
                     NETNAME=,SBUFNO=,SECSTAT=,STAXID=,STKNUM=,        C
                     UNLOAD=

Required: label,CNCNAME,CNCTYPE=EX
Defaults: BUFNO=2,BUFPOOL=YES,BUFSIZ=256,
          NETNAME=BLANKS,SBUFNO=0,SECSTAT=FF,
          STAXID=S00000,STKNUM=2,UNLOAD=NO
```

label (required)
The SNA network name. Recommended names are $NETx, where *x* is the PU number (1 – 4).

CNCNAME (required)
This operand specifies a field containing the name of the NDU SDLC device configuration record. The name must consist of 1 to 8 alphanumeric characters.

CNCTYPE (required)
Specifies to use the extended data link manager support in SNA. When you specify CNCTYPE = EX, SNA uses the extended data link manager support that allows the user to use shared SDLC support available in EDX Version 6.

BUFNO
The number of PU buffers to be allocated as free pool buffers at network activation time. This is the number of buffers that will exist in addition to those allocated for each LU at network generation time. The minimum is 2 and the default is 2.

BUFPOOL
This operand specifies whether you use buffer pooling or session buffering for the PU.

YES (default) You use buffer pooling. All buffers are shared between the LUs and the PU.

NO You use session buffering. Buffers are not shared among LUs. The PU and the LUs each have separate buffer areas that are used exclusively by the LU or the PU.

BUFSIZ
Specifies the size of the PU and LU buffers. The BUFSIZ value specified must be at least 9 bytes less than the IMAX field specified on the SDLC device configuration record, as defined through the Network Definition Utility. This buffer length is the length of the largest request/response unit (RU) to be sent out or the largest RU segment received. This size determines the size of all buffers allocated at network generation time. The minimum is 50 (to allow receipt of a minimum length BIND). The default is BUFSIZ = 256.

NETNAME
This operand specifies a field containing the network name (1 to 8 characters) of this PU. If not specified, the network name of this PU defaults to blanks.

SBUFNO
This operand specifies the number of buffers (0−253) allowed for concurrent send operations.

If buffer pooling is specified (BUFPOOL=YES), it is recommended that you enter a value for the SBUFNO parameter. If you specify a value other than 0, then you must also specify SENDBUF=0 and RECVBUF=0 on all your SNALU macros. If you specify the SBUFNO as 0, then at least one of your SNALU macros must specify SENDBUF with a value of at least 2.

If session buffering is specified (BUFPOOL=NO), then you must specify SBUFNO=0.

For either buffer pooling or session buffering, the minimum SBUFNO is 0 and the default is 0.

SECSTAT
Specifies the hexadecimal value to be used as the secondary station address to use for this PU. Valid values are from X'01' to X'FF' A value of X'FF' indicates that the secondary station address used is the one jumpered on the SDLC Attachment card (hardware address recognition). If SECSTAT is not coded, then the default value X'FF' is used. This parameter is only valid when CNCTYPE=EX.

Notes:

1. The SECSTAT value must match one of the secondary station addresses (STNx) specified on the SDLC station configuration record.
2. For hardware address recognition, the SDLC station configuration record STN1 parameter must be specified as X'FF'. STN2, STN3, and STN4 must not be specified.
3. If using shared link support, then more than one SDLC station (SNA PU) can use the same SDLC attachment and line.

> **Warning:** You cannot use hardware address recognition if you plan to have two or more SNA PUs or SNA and APPC share the same SDLC attachment and SDLC line to the host.

STAXID
The format and unique ID to be sent in response to an XID command to this station. This operand is for a switched line.

Ln Long format. Unique ID is *n*.

Sn Short format. Unique ID is *n*.

The value of *n* may be any number expressed as 5 hexadecimal digits. This ID should be unique among all Series/1 systems on the SNA network.

The format of the data in the 6-character field is as follows. The first character is expected to be an EBCDIC L or S. Any other value defaults to S. The next 5 characters are used as the unique ID (n). If not specified, the default is the short format, station=S00000.

STKNUM
Specifies the number of stacks the PU should reserve for use by the LUs. All stacks remain in a pool until you issue a request for an LU (NETGET, NETINIT, and so on). You must specify a value from 2 to 96 for STKNUM.

For duplex sessions, it is recommended that you specify two stacks for each defined LU plus one additional stack. For example, if you have defined two LUs, you would specify a total of five stacks.

For half-duplex sessions, it is recommended that you specify a number based on the number of defined LUs plus one. For example, if you have defined two LUs, you would specify a total of three stacks.

The default is STKNUM=2.

UNLOAD
Determines if $NETx is unloaded.

YES	Unload $NETx on attention **SNADACT/PUDACT**, if the host sends **DISCONNECT**, or if a permanent SDLC error condition occurs.
NO (default)	$NETx deactivates and unloads, then reloads and reactivates if you receive a **DISCONNECT** from the host.
SNADACT	$NETx deactivates and unloads, then reloads and reactivates if you receive a **DISCONNECT** from the host or if a permanent SDLC error condition occurs.

See "PU Reactivation" on page 3-12 and "Shared SDLC Error Return Codes" on page 3-29.

Note: If you enter the operator command **SNADACT**, the $SNA and $NETx programs are unloaded. To unload $SNA, you must use the **SNADACT** operator command.

Warning: For SNA Version 2.1, $C $SNA or $C $NETx (where x is the number of the PU) commands are *not* allowed.

SNAPU Parameter Considerations

Prior to redefining the default values for SNAPU parameters, you should review the following topics to determine which parameter values to specify when you define a PU.

PU Buffer Considerations

Use the BUFSIZ parameter to specify the physical size of the PU buffer (request unit). The size specified is the size of the largest acceptable request unit per frame. This value should be 9 less (6 bytes for the transmission header and 3 bytes for the request header) than the IMAX field specified on the SDLC device configuration record as defined through NDU. The IMAX field on the SDLC device configuration record should match the MAXDATA parameter for the PU in the NCP definitions.

Buffer Pooling

In buffer pool management (BUFPOOL=YES), SNA allocates a buffer pool when it starts a PU. During a session, you obtain buffers by issuing send and receive operations. The system waits until a buffer is available and then completes the send operation. The maximum number of outstanding send buffers is the value you specify for SBUFNO. If SBUFNO=0, SNA uses the sum of the values specified on SENDBUF for all LUs. SBUFNO does not define more buffers (only the limit); SENDBUF creates more buffers.

You can use send buffers for receive operations. All buffers defined are available for receive operations, but the number of buffers available for send operations is limited by SBUFNO or the sum of the SENDBUFs.

See Appendix I, "SNA Buffer Management" for more information.

Session Buffering

In session buffer management (BUFPOOL=NO), the system allocates send and receive buffers per session when starting a PU. These buffers are available only for that specific session. If there is no buffer available for a send operation, the session waits for one. When there is no buffer available for a receive operation, SNA support terminates the session.

Note: You must specify at least two send buffers and two receive buffers for each LU when using session buffering. Buffers are NOT shared between LUs.

See Appendix I, "SNA Buffer Management" for more information.

Segmented Message Buffer Considerations

When the Series/1 receives segmented messages, the number of segments in a message must not exceed the number of available buffers. SNA cannot free any buffers that contain message segments until receipt of the last segment of the message. If the number of segments exceeds the number of available buffers, SNA waits for an available buffer. The only way you can recover is to deactivate the network using the SNADACT operator command or by deactivating the PU at the host using the FORCE option.

See Appendix I, "SNA Buffer Management" for more information.

Switched Line Considerations

When you define a PU on a switched line, the NCP usually defines a unique ID (called the SDLC station identifier) for that PU within the network. When the connection is made, the NCP requests the station ID of the contacted PU and verifies it with the defined value for that PU in the NCP definitions. This allows the NCP to verify that the contacted PU is allowed to participate in the network. The STAXID parameter identifies the station ID for the PU, which must match the user-defined part of the station ID in the NCP. The PU block number or the machine type identifier for the Series/1 is X'021'

PU Reactivation

If your Series/1 is unattended, you can eliminate the need for operator commands (PUACT) or EDL instructions (NETPACT) by defining the SNAPU UNLOAD parameter as UNLOAD=SNADACT. The value you specify for UNLOAD determines whether a PU reactivates. A PU deactivates because:

- The operator enters a PUDACT or SNADACT command.

- The system detects a permanent hardware error on the SDLC link.
- The host sends an SDLC DISCONNECT command.

Take action, depending on the UNLOAD parameter, as follows:

Reason for deactivation	UNLOAD=YES	UNLOAD=NO	UNLOAD=SNADACT
PUDACT/ SNADACT command	PU deactivates and unloads	PU deactivates and unloads	PU deactivates and unloads
SDLC hardware error	PU deactivates and unloads	PU deactivates and unloads	PU deactivates and unloads, reloads and reactivates
SDLC DISCONNECT command	PU deactivates and unloads	PU deactivates and unloads, reloads and reactivates	PU deactivates and unloads, reloads and reactivates

Concurrent Requests

At any one time, many SNA instruction requests can exist for a PU. Each request requires servicing of a PU stack. You specify the number of stacks with the STKNUM operand. The amount of storage required depends on the value you specify. If you decrease the value, SNA can issue more requests than there are stacks, forcing some requests to wait until previous requests finish and the PU releases the stack. This decreases performance.

Each LU used for a duplex session needs two stacks defined (one for sending data, one for receiving data). Each LU used for a half-duplex session needs one stack. After computing the total number of stacks required, then add one more stack for session termination. The actual number of stacks needed is usually much less. Not all LUs are in session simultaneously, and an LU rarely has more than one operation outstanding at a time, unless it is duplex.

The minimum number of stacks should be one more than the maximum number of concurrent sessions. The maximum number of concurrent sessions cannot be greater than the number of defined LUs. You can define more stacks for performance reasons and for duplex LUs.

Defining SDLC Using the Network Definition Utility

For using the shared SDLC support with your SNA network, you must use the Network Definition Utility to define that support. The Network Definition Utility creates the configuration records containing the parameters that are used during network activation. For more information on using the Network Definition Utility to define your shared SDLC support, refer to the *Network Definition Utility Guide.*

Since the shared SDLC uses the Network Definition Utility to define the Secondary SDLC device characteristics, SDLC-related parameters on the SNAPU statement are not valid or have a new purpose. Figure 3-2 on page 3-14 illustrates changes to the SNAPU parameters and corresponding NDU SDLC configuration record fields.

This can be used to relate SNA considerations (described in Chapter 2, "Installing and Defining the SNA Network for Use with SDLC") to shared SDLC parameters.

SNAPU Parameters	Shared SDLC Line Characteristics Parameters	Shared SDLC Device Configuration Record Parameters	Shared SDLC Station Configuration Record Parameters	SNAPU Coding Status
ADDRESS		DVAD		Not valid
BUFNO		RBFN, TBFN		See Note 1
BUFPOOL				No change
BUFSIZ		IMAX		See Note 2
CNCNAME		NAME		See Note 3
CNCTYPE	CONN			See Note 4
DCBNO		See Note 5		Not valid
ENCODE	ENCD			Not valid
FRTPOLL			ICTL	Not valid
NETNAME				No change
NORING		See Note 6		Not valid
PAD	PADC			Not valid
RATE	RATE			Not valid
RETRY			CRTL	Not valid
SBUFNO				No change
SECSTAT			STNx	See Note 7
STAXID				No change
STKNUM				No change
THRESH				Not valid
TOCTS	TCTS			Not valid
TODSR	TDSR			Not valid
TODTR		See Note 6		Not valid
TOHLA				Not valid
TONPR	TNPR			Not valid
UNLOAD				No change

Figure 3-2. SNAPU Parameter Changes and Corresponding SDLC Configuration Fields

Note:

1. For SNA with SDLC support, the BUFNO parameter specified the number of SDLC buffers to be shared by the SDLC support, the LUs, and the PU. For SNA with shared SDLC support, the BUFNO parameter specifies the number of

LU and PU buffers. The number of SDLC buffers are specified by the RBFN and TBFN parameters on the SDLC device configuration record.

2. BUFSIZ still specifies the size of the PU and LU buffers. The IMAX field on the SDLC device configuration record specifies the size of the SDLC buffers. The BUFSIZ specified must be at least 9 bytes less than the IMAX field specified on the SDLC device configuration record.
3. The CNCNAME parameter is required on the SNAPU statement to specify the name of the NDU SDLC device configuration record. The CNCNAME parameter field must match the NAME field on the SDLC device configuration record.
4. To use the shared SDLC support, specify CNCTYPE=EX on the SNAPU statement. The EX designates extended data link manager support; namely, the support in SNA that allows the user to use shared SDLC support. The CONN parameter on the SDLC line characteristics record specifies the type of physical line connection.
5. The shared SDLC support requires the value for MAXDCB on the EXIODEV statement as 16. The MAXOUT parameter of the GROUP, LINE, or PU macro for the NCP generation should be 7.
6. The TODTR and NORING SNAPU parameters do not have a shared SDLC option. The data terminal ready and NORING values may or may not be jumpered on the feature card based on the type of physical line connection. On the SDLC device configuration, the following CONN types define the data terminal ready and NORING jumper requirements.

   PP For a permanent point-to-point or multipoint connection on a 2090 leased or nonswitched line, the data-terminal-ready (DTR) jumper (J3-12) and the NORING jumper (J3-15) should be installed on the transmit adapter. For a 2080 attachment card, no jumpers are required.

   SW For a 2090 switched connection with outgoing dial calls or for incoming manual-answer calls, the data-terminal-ready (DTR) jumper (J3-12) is removed from the transmit adapter. The NORING jumper (J3-15) should be installed on the transmit adapter. This connection point is not supported for a 2080 attachment card.

   AA For a 2090 switched connection with automatic answer of incoming calls, the data-terminal-ready (DTR) jumper (J3-12) and the NORING jumpers (J3-15) are removed from the transmit adapter. This connection point is not supported for a 2080 attachment card.
7. The SECSTAT value must match the STNx parameter specified on the SDLC station configuration record.

Considerations for SDLC Characteristics

Consider the following while determining SDLC characteristics:

- Modem rate
- Line speed
- Timeout values
- Retry count
- Frame window sizes
- SDLC device names and addresses
- Secondary station addresses.

Determining Modem Rate

The modem rate determines whether your modem runs at half capacity or full capacity. The modem rate you specify on the RATE field of the SDLC device configuration record must match the actual rate set on the device.

Determining Line Speed

All line speeds supported by the SDLC #2080 and #2090 attachments are supported by shared SDLC Note that as the line speed increases, SDLC requires more processor time to service the line. See Chapter 1, "Introduction" for a discussion of shared SDLC and the interfaces and line speeds supported for each feature card.

To determine the line speed, refer to your modem specifications.

Determining Timeouts

Refer to the documentation for your hardware for more information about what timeout values you should set. Refer to the *Network Definition Utility Guide* for an explanation of the timeout parameters you must set.

Determining Retry Count

For each secondary SDLC station, you define a frame retransmission limit, which represents the number of times you want to attempt retransmission of unacknowledged frames. The RTLn parameter is used with the nonproductive receive timeout parameter (TNPR) on the SDLC station configuration record. If this retransmission limit is exceeded, a message is displayed to indicate possible line or station problems.

Frame Window Sizes

The frame window size specifies the maximum number of I-frames that may be sent by your station before acknowledgment is received from the associated station. Specifying a large value will improve line utilization provided that the associated station can process the incoming frames successfully. If the associated station is limited by storage or processing speed, reducing the value will result in fewer frame retransmissions. This is specified by the WSZn parameter on the SDLC station configuration record.

Determining SDLC Device Names and Addresses

Set a naming convention for naming SDLC devices. The device name should reflect the name of the line it is associated with. For example, the SDLC device attached to the first PU ($NET1) could be called NETSDLC1. The device address is the hardware address of the SDLC device. The SNAPU CNCNAME parameter must match the NAME parameter on the SDLC line characteristics configuration record.

For information about defining your SDLC devices, refer to the *Network Definition Utility Guide*.

Determining the SDLC Station Configuration Record Name

The SDLC station configuration record contains the information needed to define a secondary SDLC station.

Set a naming convention for your station configuration record names. The name should reflect the name of the device to which it is associated. For example, the station configuration record for device NETSDLC1 could be called NETSTAT1 or SDLC1STA.

Determining Secondary Station Addresses

For each PU, you must specify the SECSTAT parameter on the SNAPU statement unless you want the default SECSTAT = FF for hardware station address recognition. SECSTAT = FF specifies that shared SDLC should use hardware station address recognition. Hardware address recognition means that the shared SDLC should use the secondary station address jumpered on the card. The Network Definition Utility station configuration record must have STN1 = FF. STN2, STN3, and STN4 must not be specified.

Specifying anything other than "FF" specifies the secondary station address for shared SDLC to use. For example, no secondary station has been jumpered on the SDLC card. The SNAPU statement has SECSTAT = C1. The Network Definition Utility SDLC station configuration record must have STN1, STN2, STN3, or STN4 coded as C1.

Specify the secondary station address in the SDLC station configuration record. For more information, refer to the *Network Definition Utility Guide.*

SNALU Configuration Statement

The SNALU configuration statement generates an LU control block. You need a SNALU statement for each LU defined for this PU.

Prior to redefining the default values of this control block, you should consider the number of LUs to be specified for the PU interfacing with the network. This is discussed in "SNALU Parameter Considerations" on page 3-18.

The following are the syntax and parameter descriptions for the SNALU configuration statement:

```
label      SNALU     LU=,CTE=,END=,HOLDBND=,                          C
                     MSGPRIO=,RECVBUF=,SENDBUF=

Required: LU
Defaults: CTE=value of SENDBUF or CTE=2 if SENDBUF=0,
          END=NO,HOLDBND=NO,MSGPRIO=255,RECVBUF=0,SENDBUF=0
```

LU (required)
The LU number for this PU in the Series/1 to be used for the session when SNA issues NETINIT. This required parameter must be a decimal value (1 to 32), which must match the LOCADDR statement on the VTAM/NCP generation.

CTE
Specifies the maximum number (1 to the maximum available storage) of correlation table entries (CTEs) to allocate for this LU. If you do not specify a value, the value is the number of send buffers (SENDBUF =) allocated for this session. If you do not specify a value and SENDBUF = 0, CTE defaults to 2. See "Correlation Tables" on page 13-7 for detailed information on correlation table entries.

END
Specifies whether this is the last LU for the PU.

YES	Means this is the last LU for the PU.
NO (default)	Means this is not the last LU for the PU.

HOLDBND
Specifies whether SNA should reject or hold a BIND received before a NETINIT is issued for the LU.

YES	SNA holds the BIND and processes it upon receiving a NETINIT request with ACQUIRE=NO for the LU. If you issue NETINIT ACQUIRE=YES, SNA rejects the BIND received and sends INITSELF to request a new bind.
NO (default)	If a NETINIT has not been issued, SNA rejects the BIND and returns 0801 sense code.

MSGPRIO
Specifies the priority of outbound messages for this LU. You can specify which message the extended data link manager support within SNA should service first if a bottleneck occurs. You must specify a value from 1 to 255. The default is MSGPRIO=255.

Note: If message prioritization is not important, then use the default value of 255 so as not to degrade overall performance.

RECVBUF
A decimal value that specifies the number used to allocate buffers for this particular LU at PU generation time. If you require session buffering (BUFPOOL=NO on the SNAPU statement), RECVBUF also represents the maximum number of outstanding inbound messages allowed for the specific LU. In this case, outstanding inbound messages refer to those for which your application has not issued NETGETs, but which SNA received anyway. If the number of outstanding inbound messages increases beyond the maximum allowed, the session ends. If you specify session buffering , RECVBUF must be at least 2. If you do not specify a value, RECVBUF defaults to zero.

SENDBUF
A decimal value that specifies the number used to allocate buffers for the LU at PU activation time. If you require session buffering (BUFPOOL=NO on the SNAPU statement), SENDBUF also represents the maximum number of outstanding outbound messages allowed for this specific LU. In this case, outstanding messages are those waiting to be transmitted by the extended data link manager support within SNA. If you specify session buffering, SENDBUF must be at least 2. If you do not specify a value, SENDBUF defaults to zero.

SNALU Parameter Considerations

When defining each PU, you must specify the number of LUs in the system. At PU activation, the SSCP attempts to establish a session with each LU of the Series/1.

In general, the number of LUs defined to the Series/1 should be the same as the number of LUs defined to the host for this cluster controller:

- If you define more LUs for any PU to the Series/1 than to the host, the SSCP is not aware of the extra Series/1 LUs. (For example, if you define 15 LUs to the Series/1 and 10 LUs to the host, the SSCP attempts to establish only 10 sessions.) In this case, the storage associated with the unused LUs defined to the Series/1 is not utilized. (See "Storage Requirements for SNA with Shared SDLC Support" on page 3-30 for information on LU storage sizes.)
- If you define fewer LUs for any PU to the Series/1 than to the host, the SSCP attempts to establish more sessions than the Series/1 wants established. (For example, if you define 10 LUs to the Series/1 and 15 LUs to the host, the SSCP

attempts to establish 15 sessions.) In this case, the Series/1 rejects the requests for the extra LUs.

When defining LUs for the PU, you must specify END=YES on the last LU. If END=YES is not specified on the last LU, unpredictable results may occur during program execution.

Buffer Allocation

The values you specify for SENDBUF and RECVBUF define the number of buffers allocated for the LU. If BUFPOOL=NO, SNA allocates the buffers for use by that LU only. If BUFPOOL=YES, SNA adds these buffers to the PU buffer pool for use by any LU in the network. If you specify SBUFNO as a non-zero value on the SNAPU statement, you must specify SENDBUF and RECVBUF as zero. Otherwise, the sum of SENDBUF for each of the LUs is the maximum number of outstanding send operations allowed for the PU when buffer pooling.

See Appendix I, "SNA Buffer Management" for more information.

Disposition of BIND Requests

If an LU receives a BIND request before you issue a NETINIT for the LU, the HOLDBND parameter determines the disposition of the BIND. If HOLDBND=NO, SNA rejects the BIND with a 0801 sense code (resource not available). If HOLDBND=YES, SNA holds the BIND request and does not send a response until you issue a NETINIT for the LU. Use HOLDBND only for host-initiated sessions.

The disadvantage of specifying HOLDBND=YES is that SNA can leave a large number of BIND requests outstanding in the network. You must specify the maximum number of outstanding BINDs in the VTAM START macro ITLIM parameter (refer to the *VTAM Installation and Resource Definition* manual). If the ITLIM limit is exceeded, VTAM stops servicing user requests.

Prioritizing Messages

SNA interfaces with the extended data link manager support by a SEND queue. Each PU has a separate queue that handles messages for that PU. SNA orders the queue by priority, with 0 the highest priority and 255 the lowest. SNA reserves 0 priority for PU messages. Assigning high priority to LUs with interactive sessions and low priorities to batch sessions allows SNA to send the interactive messages first if a bottleneck occurs in the extended data link manager support.

To maintain correct SNA architecture, SNA places a message from an LU after all other messages from that LU that are still on the send queue, even if the priority is higher.

Note: If message prioritization is not important to an LU, then use the default value of 255 so as not to degrade overall performance.

SNA Sharing a Link with APPC or Another SNA PU

Shared link support allows SNA to share an SDLC link with another SNA PU or APPC. By sharing your SDLC links, you reduce your hardware costs. However, for performance reasons, you may require multiple SDLC links to support required data traffic.

SNA Sharing a Link with APPC

If you define your APPC node as one or more secondary stations on a nonswitched SDLC link, SNA may also define one or more of its PUs for the same link as long as unique secondary station addresses are assigned. The shared SDLC support will route messages to and from the appropriate APPC or SNA PU. You must use the secondary station address as specified on the NDU SDLC station configuration record to share an SDLC link between APPC and SNA. You must not specify any secondary station address as X'FF'.

To use the shared link support, you must choose unique secondary station addresses and define them in the NDU SDLC station configuration record for the shared SDLC device. The same SDLC device name will be used by APPC and SNA. The SDLC device name in SNA is specified via the SNAPU CNCNAME parameter. Refer to the *Network Definition Utility Guide* for an example of using the shared link support in shared SDLC. Figure 3-3 and Figure 3-4 illustrate the relationship of one APPC PU and one SNA PU without and with shared link, respectively.

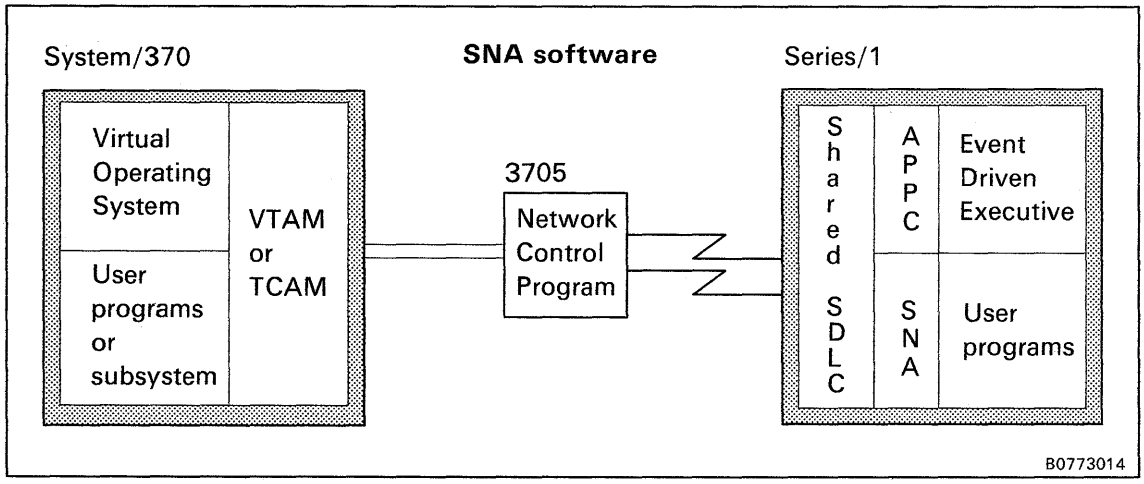


Figure 3-3. SNA and APPC without Shared Link. (SNA and APPC each have 1 PU.)

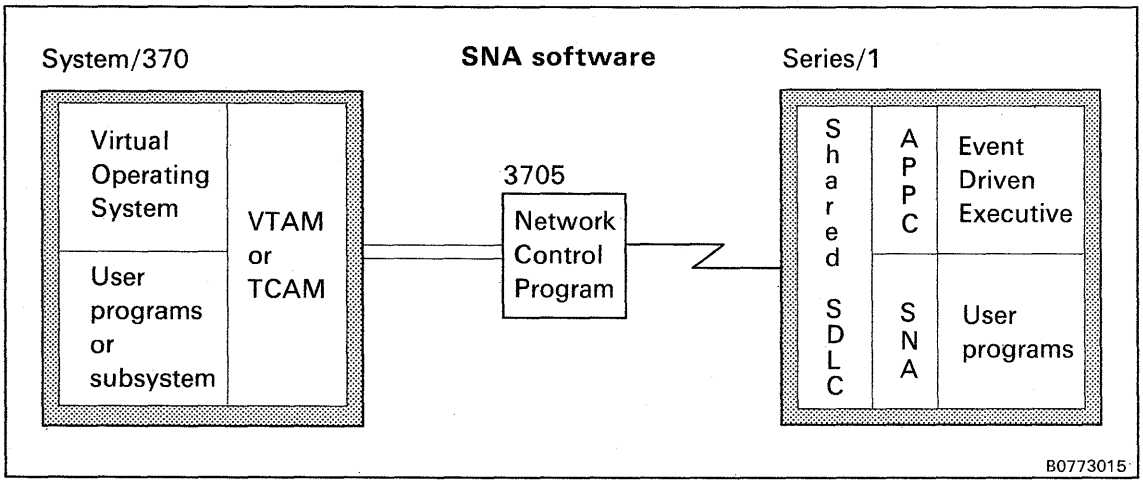


Figure 3-4. SNA and APPC with Shared Link (SNA and APPC each have 1 PU.)

SNA Sharing a Link with Another SNA PU

If you use a nonswitched SDLC link, SNA may define one or more of its PUs for the same link as long as unique secondary station addresses are assigned. Figure 3-5 and Figure 3-6 show how SNA would be without and with shared link support, respectively.

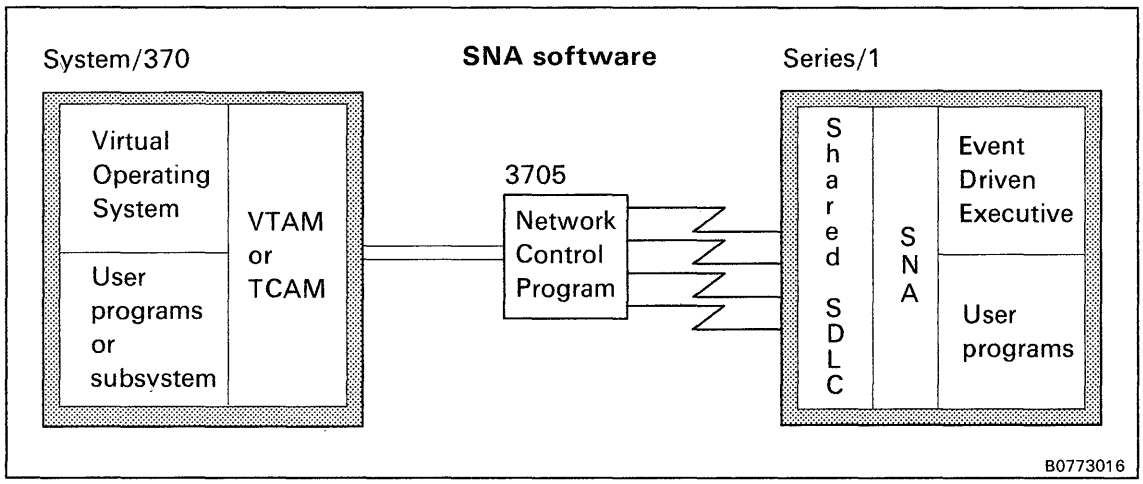


Figure 3-5. SNA with 4 PUs Not Using Shared Link

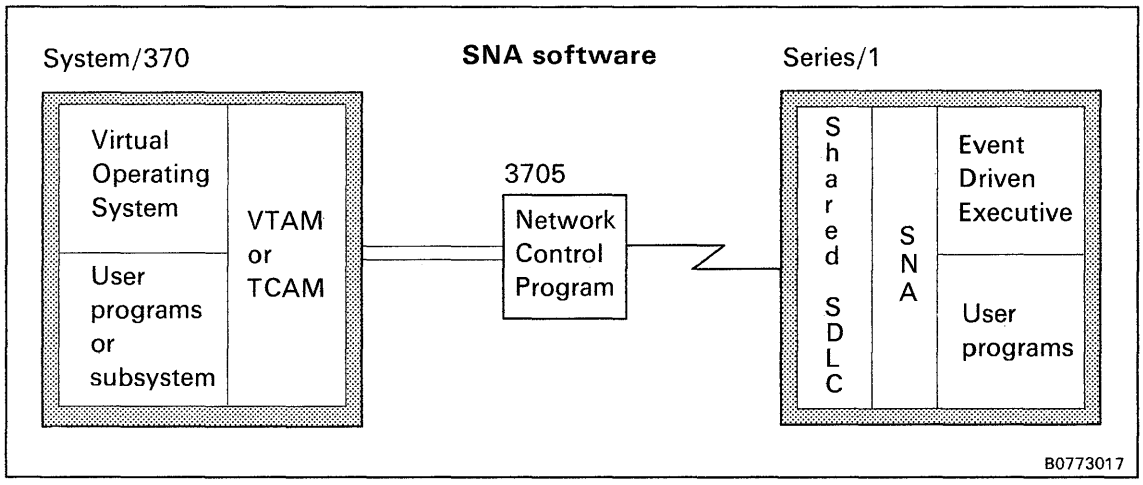


Figure 3-6. SNA with 4 PUs Using Shared Link

To define shared SDLC links, define unique secondary station addresses on the SDLC station configuration record for the STNx parameters. When editing the $SNADEFx data sets for the PUs that will use shared link, code the CNCNAME parameter with the same SDLC device configuration name. Code the SECSTAT parameter in each of the $SNADEFx data sets with a unique STNx parameter.

For example, PU#1 and PU#2 will use the shared link support on a nonswitched SDLC link. Assume on the SDLC station configuration record that STN1 = C1 and STN2 = C2. $SNADEF1 would then appear as follows:

```
NET1     CSECT ,                  DEFINE THE NETWORK
         SNAPU CNCTYPE=EX,        CONNECTION TYPE                        X
               CNCNAME=NETSDLC1,  NAME OF SDLC CONFIGURATION RECORD      X
               UNLOAD=YES,        UNLOAD $SNA IF DISC/PERM ERROR         X
               BUFSIZ=256,        SIZE OF LARGEST USER MESSAGE           X
               BUFNO=2,           NUMBER OF BUFFERS AT ACTIVATION        X
               STAXID=S00000,     ID TO SEND IN RESPONSE TO XID CMD      X
               NETNAME=NET1,      NETWORK NAME THIS CONNECTION           X
               BUFPOOL=YES,       BUFFER POOLING                         X
               SECSTAT=C1         SECONDARY STATION ADDRESS              X
               STKNUM=2,          NUMBER OF STACKS FOR LUS               X
               SBUFNO=0           CONCURRENT SEND BUFFERS
         EJECT
         SNALU LU=1,              LOGICAL UNIT NUMBER                    X
               SENDBUF=2,         NUMBER OF SEND BUFFERS FOR LU          X
               RECVBUF=2,         NUMBER OF RECEIVE BUFFERS FOR LU       X
               CTE=2,             NUMBER OF CORRELATION TABLE ENTRIES    X
               HOLDBND=NO,        HOLD BIND UNTIL NETINIT ISSUED         X
               END=YES            LAST LOGICAL UNIT TO GENERATE
         END
```

$SNADEF2 would then appear as follows:

```
NET2     CSECT ,                  DEFINE THE NETWORK
         SNAPU CNCTYPE=EX,        CONNECTION TYPE                        X
               CNCNAME=NETSDLC1,  NAME OF SDLC CONFIGURATION RECORD      X
               UNLOAD=YES,        UNLOAD $SNA IF DISC/PERM ERROR         X
               BUFSIZ=256,        SIZE OF LARGEST USER MESSAGE           X
               BUFNO=2,           NUMBER OF BUFFERS AT ACTIVATION        X
               STAXID=S00000,     ID TO SEND IN RESPONSE TO XID CMD      X
               NETNAME=NET2,      NETWORK NAME THIS CONNECTION           X
               BUFPOOL=YES,       BUFFER POOLING                         X
               SECSTAT=C2         SECONDARY STATION ADDRESS              X
               STKNUM=2,          NUMBER OF STACKS FOR LUS               X
               SBUFNO=0           CONCURRENT SEND BUFFERS
         EJECT
         SNALU LU=1,              LOGICAL UNIT NUMBER                    X
               SENDBUF=2,         NUMBER OF SEND BUFFERS FOR LU          X
               RECVBUF=2,         NUMBER OF RECEIVE BUFFERS FOR LU       X
               CTE=2,             NUMBER OF CORRELATION TABLE ENTRIES    X
               HOLDBND=NO,        HOLD BIND UNTIL NETINIT ISSUED         X
               END=YES            LAST LOGICAL UNIT TO GENERATE
         END
```

Managing Storage for Shared SDLC

The shared SDLC support uses transient segments to manage storage efficiently. A transient segment is a 512-byte group of instructions loaded from disk as needed. A transient segment is similar to an EDX overlay segment with two additional features. Transient segments are reentrant and refreshable blocks of code. Reentrant code is a group of instructions that can be executed simultaneously by more than one task in the same partition. Refreshable code is a group of instructions whose execution can be suspended and later restarted. When the transient segment is "refreshed" or loaded again from the disk, the execution continues normally.

The following sections explain how to allocate transient slots and how to customize the number of transient slots for your operating environment.

Allocating the Transient Slots

A transient slot is a 512-byte storage area into which transient segments are loaded. A transient slot contains one transient segment at a time, but transient segments are loaded into the slot as they are needed. When all of the slots are in use, the execution of one transient segment can be suspended and another loaded from disk as it is required.

You control the number of transient slots available to shared SDLC by adjusting the amount of program storage allocated for the resident programs. When the shared SDLC support is installed, the $CSDLC program has 1024 bytes of program storage. A minimum of 1024 bytes is required but you can increase the program storage to a maximum of 16384 bytes. Therefore, you must have at least two transient slots, but you can allocate up to 32 transient slots.

The following procedure allocates 2048 bytes of storage for shared SDLC, which will contain four transient slots. The amount of program storage you should allocate depends upon your operating environment, but the number you choose should be a multiple of 512 bytes. (Each transient slot requires 512 bytes.) If you allocate a large number of transient slots, your performance will improve because fewer transient segments will wait to be loaded. However, less storage will be available to execute programs. For most users, 1024 bytes will be sufficient for shared SDLC transients.

The example assumes the shared SDLC support was copied to volume EDX002.

1. Press the attention key.
2. Type in **$L $DISKUT2**.
3. Press the enter key. (The enter key must be pressed for the Series/1 to read the information you have typed on the screen. For the rest of this procedure, the instruction to "enter" a command includes pressing the enter key.)

```
>$L $DISKUT2
LOADING $DISKUT2       nnP, hh:mm:ss, LP= xxxx, PART= yy


$DISKUT2 - DATA SET MGMT. UTILITY II

USING VOLUME EDX002

COMMAND (?):  SS $CSDLC

OLD STORAGE SIZE WAS       1024
ENTER NEW STORAGE SIZE IN BYTES 2048

NEW SIZE WILL BE 2048

OK TO CONTINUE (Y/N)? Y

COMMAND (?): EN

$DISKUT2 ENDED AT hh:mm:ss
```

Customizing the Number of Transient Slots

You can customize the number of transient slots available to shared SDLC support for your operating environment. By increasing the number of transient slots, you reduce the number of times the system must wait to load a transient segment. However, you also decrease the amount of storage available for program execution.

The optimal number of transient slots to allocate depends upon your operating environment. At least two transient slots (1024 bytes) are required, but you can allocate a maximum of 32 transient slots (16384 bytes). You should consider the following questions as you customize the number of transient slots for your network:

- How many tasks (PUs) use shared SDLC support concurrently?
- Is maintaining available storage more important than improving performance for your network?

Shared SDLC has an operator command to display the number of calls to transients, the number of times a transient was loaded, and the number of times a transient waited to be loaded. These statistics will help you to "tune" the number of transient slots required by the shared SDLC support for your system.

The STRNSTAT attention list command displays the statistics gathered by the shared SDLC services from the time $CSDLC was loaded.

Note: STRNSTAT is a $CSDLC attention list command, not a $SNA attention list command. See "Display SDLC Transient Statistics" on page 3-26 for more information on the STRNSTAT operator command.

If a high percentage of calls for transient segments require that those segments be loaded or if a high percentage of transient segments must wait to be loaded into storage, then increase the number of transient slots available. If shared SDLC support never waits for an available transient slot when loading transients, you can decrease the number of transient slots available.

Displaying SDLC Statistical Information

You can display statistical information about the shared SDLC support in your network. This information is collected automatically when shared SDLC starts. You use utilities and operator commands to display device statistics and transient information.

Displaying SDLC Device Statistics

You use the $SDLCST utility to display the statistics gathered by the shared SDLC support for a specific SDLC device or all SDLC devices activated. Both device level and station level statistics are collected and reported.

To display the SDLC device statistics, load $SDLCST into any partition. The following example assumes $SDLCST is on volume EDX002.

```
>$L $SDLCST
LOADING $SDLCST       nnP, hh:mm:ss, LP= xxxx, PART= yy


SDLC DEVICE NAME: NETSDLC1

DEVICE - NETSDLC1     ADDR -   000A     STATUS - LINE SETUP

  TOTAL REQUESTS RECEIVED              =  1
  TOTAL WAITS FOR IOXS                 =  0
  TOTAL FRAMES TRANSMITTED             =  0
  TOTAL FRAMES RECEIVED                =  0
  TOTAL I/O EXCEPTION RETRIES          =  0
  TOTAL RECEIVE OVERRUNS DETECTED      =  0
  TOTAL ABORTED FRAMES RECEIVED        =  0
  TOTAL LONG FRAMES RECEIVED           =  0
  TOTAL FRAMES RECEIVED WITH FCS ERROR =  0

LOGICAL STATION ADDR - 00C1 STATUS  - INACTIVE

$SDLCST ENDED AT hh:mm:ss
```

Refer to *Operator Commands and Utilities Reference* for more information regarding the $SDLCST utility.

Note: When using shared SDLC, use $SNADISP to get all SNA statistics.

Display SDLC Transient Statistics

To display SDLC transient statistics, enter the STRNSTAT operator command in the partition where you loaded $CSDLC. The statistics are collected from the time $CSDLC is loaded.

Note: This command will be accepted only in the partition where $CSDLC was loaded.

```
>STRNSTAT
CSVC0004 ================> TRANSIENT STATISTICS <==================
CSVC0005 RESOLVED TRANSIENT VOLUME:                      EDX002
CSVC0006 RESOLVED TRANSIENT DATA SET:                  CSDLCTRN
CSVC0009 NUMBER OF CALLS TO A TRANSIENT:                     90
CSVC0007 NUMBER OF TRANSIENT LOADS:                          60
CSVC0008 NUMBER OF WAITS FOR AN AREA TO LOAD A TRANSIENT:     0
```

Allocating a $MEMDISK Volume for the Transient Libraries

You should place the transient library data sets on the fastest disk or diskette device in your system. For the fastest execution times, place the data sets on a volume created by the $MEMDISK utility. A volume created by the $MEMDISK utility is accessed quickly because unmapped storage is used as a simulated disk device. The transient library data set is loaded each time $CSDLC is loaded; therefore, you do not need to worry about losing data when you perform an IPL.

To allocate a $MEMDISK volume for the transient libraries, you need to do the following:

Step 1 Set up your system to use unmapped storage.

Step 2 Define a volume using the $MEMDISK utility.

Step 3 Copy the transient data set to the $MEMDISK volume.

Step 4 Patch $CSDLC to refer to the volume created by the $MEMDISK utility.

Step 1 - Set Up Your System to Use Unmapped Storage

To obtain unmapped storage you must edit both the $EDXDEF and $LNKCNTL data sets using the $FSEDIT utility. Both data sets reside on volume ASMLIB. In the $EDXDEF data set, you must define the amount of mapped storage required on the SYSPARTS statement. The storage remaining is unmapped storage. In the $LNKCNTL data set, you must add an INCLUDE statement for the STORMGR module. Then you must perform a system generation to include these new definitions and support. Refer to the *Installation and System Generation Guide* for more information about defining unmapped storage and performing system generations.

Step 2 - Define a $MEMDISK Volume

The $MEMDISK utility allocates unmapped storage as a "disk device." Use the $MEMDISK utility to define a volume for your transient library data set each time you perform an IPL.

The following procedure uses the $MEMDISK utility to define a volume, MEMDSK, for the SDLC transient library.

Note: See "Utilities Examples" on page xviii for more information about the symbols used in this example.

Step 2(a). Load the $MEMDISK utility.

1. Press the attention key.
2. Enter **$L $MEMDISK**.
3. Press the enter key.

```
>$L $MEMDISK
LOADING $MEMDISK     nnP, hh:mm:ss, LP= xxxx, PART= yy

MEMDISK INITIALIZATION UTILITY

COMMAND (?):
```

Step 2(b). Initialize and allocate volume MEMDSK as a $MEMDISK volume. Respond to the $MEMDISK prompt messages as shown.

```
COMMAND (?): IV

SIZE IN RECORDS (0 TO EXIT): 210

MAXIMUM NUMBER OF DATASETS: 1

VOLUME NAME: MEMDSK

MEMDSK ALLOCATED AND INITIALIZED

COMMAND: EN
$MEMDISK ENDED AT hh:mm:ss
```

Step 3 - COPY the Transient Library Data Set to the MEMDSK Volume

Now you must copy the transient data set on EDX002 to the MEMDSK volume using the utility, $COPYUT1.

Note: $MEMDISK simulates a disk device in unmapped storage; therefore, you must copy this data set each time you perform an IPL.

The following procedure copies the SDLC transient data set to the volume MEMDSK, which was created by the $MEMDISK utility.

3(a). Load the $COPYUT1 utility.

1. Press the attention key.
2. Enter **$L $COPYUT1**.
3. Respond to the $COPYUT1 prompt messages as shown.

```
>$L $COPYUT1
LOADING $COPYUT1     nnP, hh:mm:ss, LP= xxxx, PART= yy

$COPYUT1 - DATA SET COPY UTILITY
```

```
              ***WARNING***
MEMBERS ON TARGET VOLUME WILL BE DELETED
REALLOCATION AND COPYING OF MEMBERS IS
DEPENDENT ON SUFFICIENT CONTIGUOUS SPACE

THE DEFINED SOURCE VOLUME IS EDX002   OK (Y/N)?  Y
THE DEFINED TARGET VOLUME IS EDX002   OK (Y/N)?  N MEMDSK
MEMBER WILL BE COPIED FROM EDX002 TO MEMDSK   OK(Y/N)?  Y
COMMAND (?): CM CSDLCTRN *
CSDLCTRN COMPLETE          RECORDS COPIED

COMMAND: EN

$COPYUT1 ENDED AT hh:mm:ss
```

Step 4 - Patch $CSDLC to Refer to the $MEMDISK Volume

The $CSDLC program expects all the data sets that it references to be on the volume that contains the $CSDLC program. If you had installed the data sets on another volume (EDX002, for instance), you would now have to patch the $CSDLC program so that it expected the transient library data set to be on a different volume.

The following procedure patches the volume name of the transient library data set for the SDLC support. The old volume name is EDX002. The new volume name is MEMDSK.

1. Load the $DISKUT2 utility.
   a. Press the attention key.
   b. Enter **$L $DISKUT2**.

```
>$L $DISKUT2
LOADING $DISKUT2     nnP, hh:mm:ss, LP= xxxx, PART= yy

$DISKUT2 - DATA SET MANAGEMENT. UTILITY II

USING VOLUME EDX002
```

2. Respond to the prompts as shown to direct the $CSDLC program to volume MEMDSK for its transient library data set.

```
COMMAND (?): PA
PGM OR DS NAME: $CSDLC
$CSDLC IS A PROGRAM OF HEX SIZE XXXX
ADDRESS: 0044
HOW MANY WORDS? 3
(D)EC, (E)BCDIC, OR (H)EX?: E

NOW IS:
 0044  C5C4 E7F0 F0F2                    |EDX002

ENTER DATA: MEMDSK

NEW DATA:
 0044  D4C5 D4C4 E2D2                    |MEMDSK

OK (Y/N)? Y
PATCH COMPLETE
ANOTHER PATCH (Y/N)? N

COMMAND: EN
$DISKUT2 ENDED AT hh:mm:ss
```

Shared SDLC Error Return Codes

Three categories of error return codes are associated with shared SDLC:

- Communication Common Services return codes
- Data link router return codes
- Shared SDLC return codes.

These return codes are documented in the *Messages and Codes* manual.

Storage Requirements for SNA with Shared SDLC Support

Figure 3-7 describes the storage requirements for EDX SNA Version 2.1 with shared SDLC support within a Series/1. It describes the calculations needed to estimate the storage requirements of your customized configuration. With Version 2.1 of EDX SNA, the PU control blocks for each PU can be in separate partitions, and $SNA need not be in the same partition of any PU. When $SNA is loaded, SNA allocates a 1-page program control data area with a label of **DATA** in partition 1.

The following table shows calculations for four storage estimates:

- $SNA with the extended data link manager support
- $NETx with the extended data link manager support
- Shared SDLC
- Communication Common Services.

Description	# of bytes
Storage requirements for $SNA: SNA resident code with the extended DLM support	 35,554
Storage requirements for each PU ($NETx): Resident code to support the PU PU control blocks Stack area # of stacks (STKNUM) x 348 bytes LU area (# of LUs (SNALU x 350 bytes) + (8 bytes x # of CTEs (CTE))) Buffer area (# of buffers (BUFNO + SENDBUF + RECVBUF) x buffer size (BUFSIZ + 24) Total storage for PU (round up to page boundary)	 3552 1166
Storage requirements for each SNA application: $NETCMD (see note 1) $NETPACT (see note 2)	 2664 1190

Figure 3-7. Storage Requirements

Notes:

1. Required if application issues NETINIT, NETPUT, NETGET, NETCTL, or NETTERM instructions.
2. Required if application issues NETPACT instruction.

Sample Storage Calculation

Storage requirements for $SNA:	
SNA resident code with extended DLM support	35,554 bytes
Storage requirements for each PU ($NETx):	
Resident code to support the PU	35.75K bytes
PU control blocks	1166 bytes
Stack area (2 x 348 bytes)	696 bytes
LU area (1 x (350 bytes + (8 bytes x 2)))	366 bytes
Buffer area ((3 + 2 + 2) x (256 + 24 bytes))	1960 bytes
Total storage for PU (rounded to page boundary)	7.75K bytes
Storage requirements for each SNA application	
$NETCMD	2664 bytes
$NETPACT	1190 bytes

Figure 3-8. Sample Storage Calculation

Storage Calculations for Communication Common Services: The following are the storage calculations for Communication Common Services

Name	Description	Number Required	Bytes
$CDY	Communication Common Services program	1	17,322
CDY transient slots	storage for transient slots	at least 2 slots	512 × (# of slots)

Storage Calculations for Shared SDLC and Transient Slots: When $CSDLC is loaded into the system, certain modules are loaded into storage. Also, storage is allocated for the transient slots.

Additional space is required for each SDLC device that is started ($SDLCDST program and control blocks). The following table shows the storage allocated for resident modules and transient slots when the $CSDLC program is loaded.

Name	Description	Number of Slots Required	Bytes
$CSDLC	Shared SDLC program	1	11,738
SDLC transient slots	Storage for transient modules	at least 2 slots	512 × (# of slots)

Notes:

1. When no devices are active, the total space required for the SDLC support is 19200 plus the size of the transient area.
2. You control the sizes of the transient areas but the size of each transient area is always rounded up to the next page boundary. Specify a multiple of 512 bytes for each transient area.

Storage Requirements for Each SDLC Device: The SDLC program uses storage for each device started. This storage holds the device-dependent control blocks and buffers. The following table shows the storage pool required for each SDLC device started.

Name	Description	Number Required	Bytes
$SDLCDST	SDLC device handler	1	3852
IOX	SDLC request block	(# of concurrent requests)	24 X (# of concurrent requests)
DIX	SDLC base control block	1	162
LCCB	I/O execution request	(# of stations)	40 X (# of stations)
SSCB	SDLC station control block	(# of stations)	134 X (# of stations)
	SDLC frame buffers (transmit and receive)	(# of buffers)	(frame size + 8) X (# of buffers)
DDB	SDLC device descriptor block	1	134
DCB	SDLC I/O control block	16	256

Use this formula to calculate the total storage pool allocated for an SDLC device:

```
3592 + (24 * # of IOXs) + (174 * # of stations) + (frame size + 8) * (# of buffers)
```

Note: The total storage pool size should be rounded up to the nearest 1K value and specified for the PLSZ parameter when defining the SDLC device configuration record for this device.

Chapter 4. Network Operation

When $SNA first loads, it uses the information stored in the SNAINIT data set to determine which PUs $SNA should auto-start and into which partition it should load a PU. $SNA requires that SNAINIT be available on the IPL volume of the system. Initially, SNAINIT indicates that only PU #1 should activate when $SNA first starts and that SNA may load it into any partition. You can use $FSEDIT to change SNAINIT to auto-activate other PUs or to change where $SNA loads the PUs. SNAINIT contains a definition record for each of the four possible PUs. Each definition record is 4 characters long, so the format of SNAINIT is:

PU	Column
1	1–4
2	5–8
3	9–12
4	13–16

Each definition record specifies whether the PU should auto-start and into which partition $SNA should load the PU when started. The first character of the definition record is either a **Y** or **N**. **Y** (for YES) indicates that the PU should start when $SNA first activates, while **N** (for NO) indicates that the PU should *not* auto-activate. The second and third characters of the definition record are the partition number where $SNA should load the PU. The fourth is reserved.

An example of SNAINIT follows:

Y00 N08 Y02 N00

This indicates that $SNA starts PUs #1 and #3 when first activated. $SNA can load PU #1 into the first available partition and must load PU #3 into partition 2. When PU #2 starts, $SNA loads it into partition 8; and if PU #4 starts, $SNA may load it into any partition. The initial contents of SNAINIT are as follows:

Y00 N00 N00 N00

To activate any PU that does not auto-activate and to deactivate any PU without deactivating $SNA, you can use the two attention list commands: PUACT and PUDACT. Also, SNADACT deactivates all PUs before deactivating SNA and unloading $SNA.

For SNA with SDLC support, valid values are from 0 to 8, with 0 indicating a first available partition. SDLC requires that the $NETx program be loaded into a static partition from 1 to 8. For SNA with shared SDLC support, valid values are from 0 to the last partition defined in the supervisor, with 0 indicating the first available partition. Shared SDLC allows $NETx to be loaded in any partition, dynamic or static.

Figure 4-1 and Figure 4-2 on page 4-2 illustrate partition mapping for the processors.

Partition	$SNASDLC	$NETx w/$SNASDLC	$SNAEX	$NETx w/$SNAEX	User Program
Static 1-8	Allowed	Allowed	Allowed	Allowed	Allowed
Dynamic 1-16	Allowed	Invalid	Allowed	Allowed	Allowed

Figure 4-1. Partition Mapping for Any Processor

Partition	$SNASDLC	$NETx w/$SNASDLC	$SNAEX	$NETx w/$SNAEX	User Program
Static 1-8	Allowed	Allowed	Allowed	Allowed	Allowed
Dynamic 1-8	Allowed	Invalid	Allowed	Allowed	Allowed
Static or dynamic 9-32	Allowed	Invalid	Allowed	Allowed	Allowed

Figure 4-2. Partition Mapping for 4956 Model J and K Processors with EDX Version 6

Note: When using SNA with shared SDLC support, the Communication Common Services program ($CDY) is loaded into the first available partition. The shared SDLC main program ($CSDLC) is loaded into the partition specified by the SDLC start-up configuration record. $CSDLC may be loaded into any partition, static or dynamic. The $SDLCDST programs will be loaded into a static partition (1 – 8) only. For more information, refer to the *Network Definition Utility Guide*.

When using SDLC support, $SNASDLC is renamed to $SNA. When using shared SDLC support, $SNAEX is renamed to $SNA.

Application Interfaces

You can load $SNA from your application with the EDL LOAD command. Processing is the same as if you loaded $SNA using the $L command. Additionally, the application can override the SDLC station ID defined for a specific PU or all PUs. Specify the station ID on the STAXID parameter for the SNAPU statement only for a switched line.

Each SDLC station ID requires two words. The station ID is left-justified in the 2-word area. Each PU has a separate station ID, so you need eight words to hold all four SDLC station IDs. If you do not want to override the SDLC station ID, specify zeros in the two words.

When loading $SNA with the EDL LOAD instruction, you must specify the name of the 8-word area containing the SDLC station IDs for the four PUs.

The following example shows the load instruction for $SNA. It sets the station ID for the first PU as X'12345' and the station ID for the fourth PU as X'000C1'.

```
  * LOAD SNA INTO ANY PARTITION

  * OVERRIDE STATION ID FOR PU #1 WITH X'12345'

  * OVERRIDE STATION ID FOR PU #4 WITH X'000C1'

  * LEAVE STATION IDS FOR PU #2 AND PU #3 AS THEY ARE IN THE GEN
                    *
         LOAD  $SNA,$PARMS,PART=ANY
           •
           •
           •
$PARMS   DATA  4X'12345000'       PU NUMBER 1 STATION ID
         DATA  4X'00000000'       DEFAULT PU NUMBER 2 STATION ID
         DATA  4X'00000000'       DEFAULT PU NUMBER 3 STATION ID
         DATA  4X'000C1000'       OVERRIDE PU NUMBER 4 STATION ID
```

Activating a PU - NETPACT

You can activate a PU from an application program. NETPACT starts SNA (if it was not started previously) by loading $SNA from the IPL volume. $SNA is loaded into the first available partition. Return codes inform you that NETPACT started SNA or that the PU is already active. A −1 or positive return code does not mean that the PU is active. It means only that SNA successfully loaded the PU control blocks and that the PU is attempting to activate.

To use NETPACT, you must link the SNA module $NETPACT with your application. You do not have to link $NETPACT if you do not use the NETPACT command.

If you issue a NETINIT request while the PU is activating, SNA queues the request until the PU becomes active or the activation fails. If you want to know explicitly when the PU activates, reissue the NETPACT request until you receive the "PU already active" (2) return code.

If NETPACT loaded $SNA and NETPACT receives a bad return code, all PUs are deactivated, and $SNA unloads from the system.

The following is the coding syntax and parameter descriptions of the NETPACT instruction.

```
label      NETPACT   PU=,PART#=,XID=,                              C
                     P1=,P2=,P3=

Required:  PU=
```

PU (required)
The label of a word that contains the number of the PU to activate. The value contained in this word is from 1 to 4.

PART#

The label of a 1-word area that contains the number of the partition in which to load the PU control blocks.

When used with SDLC support, the partition must be static, and this word may contain a value from 0 to 8.

When used with shared SDLC support, the partition may be static or dynamic, and this word may contain a value from 0 to the maximum number partition defined in the supervisor.

If you do not enter PART#, SNA uses the value specified for the PU in the SNAINIT data set. For valid values see Figure 4-1 on page 4-1 and Figure 4-2 on page 4-2.

Note: NETPACT checks for the value of the PART# parameter to be in the range from 0 to the maximum number partition defined in your supervisor. If the value is not in this range, NETPACT returns an error code. If you are using SDLC support (rather than shared SDLC support), you should code the PART# parameter with a value from 0 to 8, that is the value for a *static* partition. If you code a value outside the range 0 – 8 but within the valid range for your supervisor (for instance, a value of 32 if the maximum number partition defined in your supervisor is 32), NETPACT checks that the value you code for PART# is within the valid range for your supervisor, and returns no error return code. Therefore, NETPACT loads the $NETx program in the maximum number partition defined in your supervisor (in this case, 32), and the results are unpredictable.

Px

Parameter naming operands. See "Using the Parameter-Naming Operands" on page 1-11 for a further description. For NETPACT, Px corresponds to the following parameters:

P1—PU

P2—PART#

P3—XID

XID

The label of a 3-byte area that contains an SDLC station identification to override the SDLC station identification (STAXID) specified in the SNAPU macro definition. The value must be entered as 5 hexadecimal digits (left-justified). SNA ignores this parameter for a point-to-point connection.

NETPACT Return Codes

The following are the decimal return codes you can receive from a NETPACT command:

Return Code	Condition
2	PU already active.
1	PU activation successfully started and $SNA loaded into the system.
-1	PU activation successfully started.
-2	Invalid PU number.
-3	$SNA deactivation in progress.
-4	Load failed for the PU control blocks ($NETx).
-5	$SNA deactivating or not usable.
-6	Load failed for $SNA.
-7	Invalid partition number specified by NETPACT.
-8	Invalid partition number in the SNA initialization data set. (SNAINIT)
-9	Read failed for the SNA initialization data set (SNAINIT).

NETPACT Example

```
EXAMPLE OF STARTING A PU FROM AN APPLICATION

* THE PU BEING STARTED IS NUMBER 4

* LOAD THE PU INTO PARTITION #3

* OVERRIDE THE STAXID IN THE GEN WITH 000C1

      NETPACT PU=NETPU,        START PU #4                          C
              PART#=PART,      OVERRIDE THE PARTITION # IN SNAINIT  C
              XID=XIDC1        OVERRIDE THE STAXID IN THE GEN
         .
         .
         .
NETPU   DATA  F'4'             PU NUMBER 4
XIDC1   DATA  3X'000C10'       STAXID  (LAST NYBBLE IGNORED)
PART    DATA  F'3'             PARTITION NUMBER 3
```

Operator Commands

You can use operator commands to activate and deactivate a PU and to deactivate SNA. These operator commands—PUACT, PUDACT, and SNADACT—must be entered in the partition where $SNA is loaded.

Activating a PU - PUACT

The PUACT attention list command activates a PU after SNA starts. To activate a PU, enter the PUACT attention list command along with the number of the PU to be activated. If you do not supply a PU number, SNA prompts you for the PU number. SNA then tries to activate the PU for you. SNA loads the PU into the partition indicated for that PU in the SNAINIT data set.

Deactivating a PU - PUDACT

The PUDACT attention list command deactivates a PU after the PU starts. To deactivate a PU enter the PUDACT attention list command along with the number of the PU to be deactivated. If you do not supply a PU number, SNA prompts you for the PU number. SNA then tries to deactivate the PU for you by cleaning up all sessions, stopping the SDLC station, and unloading the PU from storage.

Deactivating SNA - SNADACT

The SNADACT attention list command deactivates SNA after it first deactivates all active PUs. After deactivating all PUs, SNA releases all storage obtained by $SNA and SNADACT unloads $SNA. SNADACT is the only command that unloads $SNA.

You should always use PUDACT to deactivate the PUs only. Use SNADACT to deactivate all PUs and unload $SNA. The $C operator command is not allowed and will not unload the $NETx or $SNA programs.

Examples

The following shows SNA usage of the PUACT, PUDACT, and SNADACT attention list commands. The SNAINIT data set was specified as:

```
Y00 N03 Y04 N00
```

Assume that:

- $SNA is loaded into partition 2
- No PUs can fit into partitions 1 and 2
- Only two PUs can fit into partition 3.

```
>$L $SNA
LOADING $SNA          nnnP, hh:mm:ss, LP= xxxx, PART= yy

NO SPACE FOR PROGRAM

NO SPACE FOR PROGRAM

LOADING $NET1         nnnP, hh:mm:ss, LP= xxxx, PART= 03
mm/dd/yy  hh:mm:ss    $NET1   NETWORK ACTIVATION IN PROGRESS
LOADING $NET3         nnnP, hh:mm:ss, LP= xxxx, PART= 04
mm/dd/yy  hh:mm:ss    $NET3   NETWORK ACTIVATION IN PROGRESS

> PUACT 2

LOADING $NET2         nnnP, hh:mm:ss, LP= xxxx, PART= 03
mm/dd/yy  hh:mm:ss    $NET2   NETWORK ACTIVATION IN PROGRESS

> PUACT 4

NO SPACE FOR PROGRAM

NO SPACE FOR PROGRAM

NO SPACE FOR PROGRAM

LOADING $NET4         nnnP,hh:mm:ss, LP= xxxx, PART= 04
mm/dd/yy  hh:mm:ss    $NET4   NETWORK ACTIVATION IN PROGRESS

> PUDACT 2

mm/dd/yy  hh:mm:ss    $NET2   NETWORK DEACTIVATION STARTED
mm/dd/yy  hh:mm:ss    $NET2   NETWORK DEACTIVATED

$NET2   ENDED AT hh:mm:ss

> PUDACT 4

mm/dd/yy  hh:mm:ss    $NET4   NETWORK DEACTIVATION STARTED
mm/dd/yy  hh:mm:ss    $NET4   NETWORK DEACTIVATED

$NET4   ENDED AT hh:mm:ss

> PUACT 4

NO SPACE FOR PROGRAM
```

```
NO SPACE FOR PROGRAM

LOADING $NET4          nnnP,hh:mm:ss, LP= xxxx, PART= 03
mm/dd/yy  hh:mm:ss   $NET4   NETWORK ACTIVATION IN PROGRESS

> SNADACT

mm/dd/yy  hh:mm:ss   $NET1   NETWORK  DEACTIVATION STARTED
mm/dd/yy  hh:mm:ss   $NET3   NETWORK  DEACTIVATION STARTED
mm/dd/yy  hh:mm:ss   $NET4   NETWORK  DEACTIVATION STARTED
mm/dd/yy  hh:mm:ss   $NET1   NETWORK  DEACTIVATED

$NET1   ENDED AT hh:mm:ss
mm/dd/yy  hh:mm:ss   $NET3   NETWORK  DEACTIVATED

$NET3   ENDED AT hh:mm:ss
mm/dd/yy  hh:mm:ss   $NET4   NETWORK DEACTIVATED

$NET4   ENDED AT hh:mm:ss

$SNA    ENDED AT hh:mm:ss
```

Displaying SNA Status - $SNADISP

When you load the $SNADISP program, the system responds with:

```
SNA STATUS AT mm/dd/yy       hh:mm:ss
```

If the network is not active, the following message appears:

```
$SNA PROGRAM NOT FOUND
```

If the network is active, the status of each PU is displayed. This includes the status of all the LUs associated with each PU.

If a PU is inactive, the following message appears:

```
PUxx NOT ACTIVE
```

where *xx* is the PU number (1 – 4).

SNA sets up the following header for each PU:

```
 PU    ADDR    RTRY    FLG1
----   ----    ----    ----
```

The fields are:

PU*xx*

- For SDLC, the PU device address.
- For shared SDLC, the PU number.

ADDR

- For SDLC, the storage address of the PU.
- For shared SDLC, this will be a string of dots.

RTRY For SDLC, the retry limit; any value 0 to 255 is valid. For shared SDLC, this will be a string of dots.

FLG1 This flag field is mask-bit significant. The values are:

X'8000' Link Failure Flag:

0 = No link failed
1 = A link failure occurred

X'4000' Network Activator Flag

0 = Network activated by OPEN
1 = Network activated by COMMAND

X'2000' Activate Physical Unit (ACTPU) Finite State Machine (FSM)

0 = PU reset
1 = PU active

X'1000' Immediate Control for Expedited Flow FSM

0 = Reset
1 = Pending

X'0800' Immediate Control for Normal Flow FSM

0 = Reset
1 = Pending

X'0400' Request Discontact FSM

0 = Reset, R(+/- RSP.REQDISC)
1 = Active, S(REQDISCONT)

X'0200' Request Discontact Ever Sent FSM

0 = No
1 = Yes

X'0100' Minus Response to Request Discontact Received FSM

0 = No
1 = Yes

X'0080' Pending Deactivation FSM

0 = Not pending deactivation
1 = Pending deactivation

X'0040' Discontact Received FSM

0 = No
1 = Yes

X'0020' Summary PU Service (Net Deactivation/last close) required FSM

0 = No
1 = Yes

X'0010' Set Normal Response Mode (SNRM) Received FSM

0 = No
1 = Yes

X'0008' LU Waiting for Buffer Post Flag

0 = No
1 = Yes

X'0004' Segmentation FSM

0 = State 1
1 = State 2

X'0002' Send Alert in Progress

0 = No
1 = Yes

X'0001' PU Waiting for Buffer Post

0 = No
1 = Yes

The following header is set up for the LUs:

```
LU   ADDR  EOPR  FLG1  FLG2  FLG3 ....FLAG7.....TERM.....LUSTATE
---  ----  ----  ----  ----  ---- ------------ ---- ------------
```

These fields are as follows:

LU*xx* The LU number; the valid range for *xx* is X'01' to X'20' (1 to 32 decimal).

ADDR The storage address of the LU.

EOPR The current operation in progress hardware error field. If the LUL and/or LUE does not exist, this field fills with nulls (....). The following values are valid:

X'00' NETINIT

X'01' NETPUT

X'02' NETGET

X'03' NETCTL

X'05' NETTERM

FLG1 The LUE FLAG1 field. If the LUL and/or LUE does not exist this field fills with nulls (....). This field is mask-bit significant and its values are:

X'8000' Ready to Receive (RTR)

0 = No ready to receive required
1 = Ready to receive required

X'4000' Session Event Preposted

0 = Session event not preposted
1 = Session event preposted

X'2000' End of Message Reported

0 = No
1 = Yes

X'1000' Ready to Receive Sent

0 = Ready to receive not sent
1 = Ready to receive sent

X'0800' End Bracket Received

0 = End bracket not received
1 = End bracket received

X'0400' Formatted Headers Received

0 = Formatted headers not received
1 = Formatted headers received

X'0200' Resynchronous Storage Type

0 = Disk
1 = Storage

X'0100' Resynchronous Flag

0 = No
1 = Yes (use session resynchronization)

X'0080' First NETPUT had EOT=YES

0 = 1st NETPUT had EOT=NO
1 = 1st NETPUT had EOT=YES

X'0040' Begin Bracket Received

0 = Begin bracket not received
1 = Begin bracket received

X'0020' Cancel sent

0 = Cancel not sent
1 = Cancel sent

X'0010' NETPUT bid reject reported

0 = Bid reject not reported
1 = Bid reject reported

X'0008' NETGET exhausted SNA buffer

0 = Buffer not exhausted
1 = Buffer exhausted

X'0004' LUE exists

0 = LUE does not exist
1 = LUE exists

X'0002' Repeat check loop

0 = No repeat
1 = Repeat

X'0001' Purge until end of chain

0 = Not in purge state
1 = Purge state

FLG2 The LUE Flag2 field. If the LUL and/or LUE does not exist, this field fills with nulls (....). This field is mask bit significant and its values are:

X'8000' NETINIT in progress

0 = Not in progress
1 = In progress

X'4000' NETPUT in progress

0 = Not in progress
1 = In progress

X'2000' NETGET in progress

0 = Not in progress
1 = In progress

X'1000' NETCTL in progress

0 = Not in progress
1 = In progress

X'0800' NETTERM in progress

0 = Not in progress
1 = In progress

X'0400' BID or function management data, begin bracket received

0 = Not received
1 = Received

X'0200' First NETPUT issued

0 = NETPUT not issued
1 = First NETPUT issued

X'0100' Next SNA buffer

0 = LUEGSBO refers to the first SNA buffer
1 = LUEGSBO refers to the next SNA buffer

X'0080' GET per request unit

0 = Get receives until EC or get buffer full
1 = Get receives one RU

X'0040' Session mode

0 = Half-duplex
1 = Duplex

X'0020' NETTERM hold

0 = Hold=no
1 = Hold=yes

X'0010' Definite response outstanding

0 = No
1 = Yes

X'0008' Abnormal termination

0 = Not abnormal termination
1 = Abnormal termination

X'0004' Chase sent during termination

0 = Chase not sent
1 = Chase sent

X'0002' Request shutdown sent

0 = Request shutdown not sent
1 = Request shutdown sent

X'0001' Duplex multiple element chain

0 = No
1 = Yes

FLG3 The LUS FLAG1 field. If the LUL and/or LUE does not exist, this field fills with nulls (....). This field is mask-bit significant, and its values are:

X'8000' LUS in use

0 = LUS available
1 = LUS in use

X'4000' Hold BIND when received

0 = Reject BIND if no NETINIT issued
1 = Hold BIND until NETINIT issued

X'2000' Activate LU (ACTLU) FSM

0 = Reset
1 = Active

X'1000' Immediate control for expedited flow FSM

0 = Reset
1 = Pending

X'0800' Immediate control for normal flow FSM

0 = Reset
1 = Pending

X'0400' Reserved

X'0200' TERM-SELF FSM

0 = Pending
1 = Reset

X'0100' INIT-SELF FSM

0 = Reset
1 = Pending

X'0080' Unbind received indicator

0 = Unbind not received
1 = Unbind received

X'0040' Network aborted indicator

0 = Network not aborted
1 = Network aborted

X'0020' TERM-SELF sent indicator

0 = TERM-SELF not sent
1 = TERM-SELF sent

X'0010' Unbind-Hold received indicator

0 = Unbind-hold not received
1 = Unbind-hold received

X'0008' SSCP-ID specified on NETHOST statement

0 = SSCP-ID not specified
1 = SSCP-ID specified

X'0004' Waiting on buffer available

0 = Not waiting on buffer
1 = Waiting on buffer

X'0002' Undefined

X'0001' Undefined

FLAG7 The 7-byte FSM flag field. If the LUL does not exist, this field fills with nulls (.... ....). This field is mask bit significant and the valid values are:

Word 1

X'E000' Quiesce at end of chain received FSM

000 = Reset
001 = Pending reset 2
011 = Active
100 = Pending active response
101 = Pending active quiesce
110 = Pending reset 1
010 = Reserved
111 = Reserved

X'1000' Immediate control FSM (expedited flow)

0 = Reset
1 = Pending

X'0800' Immediate control FSM (normal flow)

0 = Reset
1 = Pending

X'0400' Contention winner FSM

0 = Reset, R(EC)
1 = Pending, S(-RSP)

X'0200' Error FSM

0 = Reset, R/S(EC)
1 = Pending, R(-RSP)

X'0100' Send/receive indicator

0 = Send set by CDDRSSNN/CDDRQSNN
1 = Receive set by CDDRQRCV/CDDRSRCV

X'00E0' Quiesce at end of chain sent

000 = Reset
001 = Pending reset
011 = Active
100 = Pending active response
101 = Pending active quiesce
010 = Reserved
110 = Reserved
111 = Reserved

X'0010' Bind FM profile

0 = FMP3
1 = FMP4

X'0008' Bind TS profile

0 = TSP3
1 = TSP4

X'0007' Shutdown received FSM

000 = Reset
001 = Pend.active.response
010 = Pend.active.SHUTC
011 = Active
100 = Pending reset 1
101 = Pending reset 2
110 = Reserved
111 = Reserved

Word 2

X'8000' Expedited outbound correlation table empty

0 = Empty
1 = Not empty

X'4000' Expedited inbound correlation table empty

0 = Empty
1 = Not empty

X'2000' Normal outbound correlation table empty

0 = Empty
1 = Not empty

X'1000' Normal inbound correlation table empty

0 = Empty
1 = Not empty

X'0800' Queue response indicator (QRI) FSM

0 = Reset
1 = Pending

X'0600' Start data traffic (SDT) FSM

00 = Reset
01 = Pend.active
10 = Pend.reset
11 = Active

X'0100' Chain send FSM

0 = Between chain
1 = In chain

X'00C0' Chain receive FSM

00 = Between chain
01 = In chain
10 = Purge
11 = Reserved

X'0020' User message

0 = No message for the user
1 = Message for the user

X'0010' Send/receive recovery signal

0 = Receive signal
1 = Send signal

X'0008' Session mode

0 = Half-duplex
1 = Duplex

X'0004' Release buffer

0 = Do not release buffer
1 = Release buffer

X'0002' Correlation Table (CT) Process

0 = Do not call CT process
1 = Call CT process

X'0001' HDX Receive (HDXREC) caller indicator

0 = Called by HDXBR routine
1 = Called by HDXREC routine

Word 3

X'8000' Alternate code

0 = Alternate code may not be used
1 = alternate code may be used

X'4000' Buffer overrun

0 = Buffer count ok
1 = Buffer count overrun

X'2000' Inbound pacing request (RQ) FSM

0 = No RQ pending
1 = RQ pending

X'1000' and X'0800' Outbound pacing FSM

0 0 = Reset Q open
0 1 = Pacing response received, Q needs driving
1 0 = Pacing request sent, Q open
1 1 = SENT.WAITING for pacing response, Q closed max count

X'0400' Activate LU (ACTLU) post

0 = No post required
1 = Post required

X'0200' Request send

0 = Not sent
1 = Sent

X'0100' STSN receive indicator

0 = Reset
1 = STSN in progress

X'0080' Buffer used

0 = Buffer not used
1 = Buffer used

X'0040' Bracket termination

0 = Rule #2
1 = Rule #1

X'0020' Wait for bind

0 = System to wait for bind
1 = User to wait for bind

X'0010' CT search (CTSRCH) exit indicator

0 = Do not exit CTSRCH
1 = Exit CTSRCH

X'0008' RU from SLU

0 = Single RU from SLU
1 = Multiple RUs from SLU

X'0006' Mask for SLU chain

00 = No response
01 = Exception response
10 = Definite response
11 = Definite or exception response

X'0001' SLU send EB mask

0 = SLU will not send EB
1 = SLU may send EB

Seventh Byte

X'8000' FM header mask

0 = No headers
1 = FM headers allowed

X'4000' Bracket test mask

0 = No brackets
1 = Brackets will be used on this session

X'3000' Extended bind check

00 = No extended bind check
01 = User does extended bind check
10 = System does extended bind check
11 = Reserved

X'0800' Brackets reset mask

0 = Brackets reset state = BETB
1 = Brackets reset state = INB

X'0400' Waiting on NSPE indicator

0 = Not waiting on NSPE
1 = Waiting on NSPE

X'0200' NETCLOSE HOLD=YES

0 = Not issued
1 = Issued

X'0100' Passthru session requested

0 = Not a passthru session
1 = Passthru session

TERM LU termination flag field. If the LUL does not exist, this field fills with nulls (....). This field is mask bit significant, and its values are as follows:

X'C000' Bind FSM states

00 = Active
01 = Pend.reset
10 = Pend.active
11 = reset

X'2000' Clear received

0 = Clear not received
1 = Clear received

X'1000' Close in progress

0 = Close not in progress
1 = Close in progress

X'0800' Activate LU (ACTLU) received

0 = ACTLU not received
1 = ACTLU received

X'0400' Deactivate LU (DACTLU) received

0 = DACTLU not received
1 = DACTLU received

X'0200' Buffer count exceeded maximum

0 = Buffer count has not exceeded maximum
1 = Buffer count exceeded maximum

X'0100' SDLC failure

0 = No SDLC failure
1 = SDLC failure

X'0080' Failure to get storage to send request

0 = Storage received
1 = Failure to get storage

X'0040' Disconnect received

0 = Disconnect not received
1 = Disconnect received

X'0020' Activate PU (ACTPU) received

0 = ACTPU not received
1 = ACTPU received

X'0010' Deactivate PU (DACTPU) received

0 = DACTPU not received
1 = DACTPU received

X'0008' Negative response to request discontact

0 = Positive response to request discontact
1 = Negative response to request discontact

X'0004' Network aborted

0 = Network not aborted
1 = network aborted

X'0002' UNBIND-HOLD received

0 = UNBIND-HOLD not received
1 = UNBIND-HOLD received

X'0001' Reserved termination flag

LUSTATE This field consists of the half-duplex flip-flop state and the bracket state. Valid values in this field are:

If bind state is ACTIVE the possible values are:

Half-Duplex Flip-Flop States

SEND.	Series/1 has the right-to-send
RCV.	Host has the right-to-send
CONT.	Either the host or the Series/1 has the right-to-send (between brackets)
ERP1.	Series/1 error recovery
PEND1.	Awaiting message or response
PEND2.	Awaiting message or response
PEND.R.	Awaiting message or response
.... (nulls)	Reserved

Bracket States

BETB	Between brackets
PEND.INB	In brackets after response
PEND.BB	Begin brackets after response
PEND.BTB.RSP.S	Between brackets after response to send
PEND.BTB.RSP.R	Between brackets after response to receive
PEND.BTB.EC.R	Between brackets after end chain received
PEND.BTB.PURGE.R	Between brackets after purge received
INB	In brackets
PEND.BTB.EC.S	Between brackets after end chain sent
PEND.BTB.PURGE.S	Between brackets after purge sent
PEND.BTB.R	Between brackets after receive
PEND.BTB.S	Between brackets after send
.... (nulls)	Reserved

If the bind state is UNBOUND, the value is:

*** UNBOUND ***

If the bind state is PEND.RESET, the value is:

*** BIND.PEND.RESET ***

If the bind state is PEND.ACTIVE, the value is:

*** BIND.PEND.ACTIVE ***

If the LUL does not exist, the value is:

*** UNBOUND ***

Chapter 5. Activating a Session - NETHOST and NETINIT

Establishing an SNA session is the process of creating a logical communications path between two LUs—your Series/1 SNA application and the host application—for subsequent data and message exchange.

The NETHOST instruction defines requirements and resources. The NETINIT instruction establishes the session.

Building a Host ID Data List - NETHOST

A host ID data list defines a Series/1 LU. The NETHOST instruction generates an assembly-time host ID data list that defines the LU requirements and some of the session resources. NETHOST is not an executable instruction; it only defines a data area.

Note: In coding NETHOST, you may require the assistance of the host system programmer, as use of some of the parameters can influence the performance of other LUs. Other parameters require the host system programmer's knowledge of SNA protocols.

The following is the coding syntax and parameter descriptions of the NETHOST instruction. The IS prefix on the parameters is an abbreviation for INIT-SELF.

```
label       NETHOST     ISAPPID=,ISMODE=,ISPASWD=,ISQUEUE=,          C
                        ISRQID=,ISUSFLD=,SSCPID=

Required:   ISAPPID,ISMODE
Defaults:   ISPASWD,ISRQID,ISUSFLD  (all default to 8 blanks)
            ISQUEUE=NO,SSCPID=000000000000
Indexable:  none
```

Note: SNA uses the IS parameters only if NETINIT ACQUIRE=YES.

ISAPPID (required)
A 1–8 character name that identifies the host user program identification (APPLID) to be used for a session. This is the name of the program as defined to VTAM. NETINIT ignores trailing blanks.

ISMODE (required)
A 1–8 character name that identifies the set of rules and protocol for a session, which the system services control point (SSCP) uses to build the CINIT request, and which SNA uses to build the BIND request sent to the Series/1 for the session. This is the logmode table ENTRY name.

ISPASWD
A 1–8 character password used to verify the identity of a Series/1 user. The default is eight blanks, which causes NETINIT to generate a null (zero length) field in the INIT-SELF command. NETINIT ignores trailing blanks.

ISQUEUE
Queue the INIT-SELF request if it cannot execute immediately. Specify one of the following:

YES	Queue the request.
NO (default)	Do not queue the request.

ISRQID
A 1 – 8 character ID of the Series/1 user initiating a request. You can use ISRQID also to establish authority to access a particular resource. The default is eight blanks, which causes NETINIT to generate a null (zero length) field in the INIT-SELF command. NETINIT ignores trailing blanks.

ISUSFLD
A 1 – 20 character string for carrying data that you specify. Network services request processors do not process this data. SNA passes the data to the primary LU (PLU). The default for ISUSFLD is eight blanks. This causes the INIT-SELF command, on the NETINIT instruction, to generate a null (zero length) field. NETINIT ignores trailing blanks.

SSCPID
The system services control point (SSCP) identification for the network to be attached. Code this operand using 0 to 12 hexadecimal digits. A 0 value specifies the session is open with any SSCP attached. You can specify any 6-byte binary value; however, to be meaningful, the bit representation must match the ID of the attached SSCP. The default is six bytes containing zeros.

NETHOST Return Codes

NETHOST does not pass return codes; however, it identifies assembly-time parameter specifications and syntax errors.

Establishing an SNA Session - NETINIT

Either the Series/1 SNA application program or the host application can initiate a request to establish a session. The SNA support handles the protocols required to establish a session for the application program.

When the Series/1 SNA application issues NETINIT ACQUIRE=YES, SNA initiates the host session. When the application issues NETINIT ACQUIRE=NO, SNA waits until the host initiates a session. In either case, the host issues a BIND command to define the protocol for the session with the Series/1.

If the conditions for session establishment between the application programs are not acceptable, the Series/1 SNA support rejects the command, and the Series/1 application receives a return code indicating NETINIT failed. If SNA support accepts the command, the host application and the SNA application are said to be "in session." SNA support can optionally send the BIND command parameters from the host to the SNA application as data.

Once the applications are in session, data and message exchange can occur. The established session remains in effect until a NETTERM instruction ends the session. You *must always* issue NETTERM to end the session if NETINIT was successful.

The following is the coding syntax and parameter descriptions of the NETINIT instruction.

```
label       NETINIT     LU=|HOLDLU=,HOSTID=,ACQUIRE=,ATTNEV=,          C
                        ERRCODE=,EXIT=,FULLDPX=,LUSWAIT=,              C
                        MSGDATA=,MSGPRIO=,PACB=,PTHRU=,                C
                        P1=,P2=,P3=,P4=,P5=,P6=,P7=,                   C
                        RDSCB=,RESYNC=,RTYPE=,SESSPRM=

Required:   LU|HOLDLU,HOSTID,PACB (only if PTHRU=YES)
Defaults:   ACQUIRE=YES,FULLDPX=NO,LUSWAIT=YES,MSGPRIO=255.
            PTHRU=NO,RESYNC=YES,RTYPE=DISK
Indexable:  none
```

LU (required if not using HOLDLU)

The label of a 1-word field containing the number of the LU and the number of the PU to be used for this session. The high-order byte of this field identifies the PU. You must set this byte to a value from 0 to 4. If you specify a value of 0, Series/1 SNA support uses PU #1 for the session; otherwise it uses the value of the PU you specify.

The low-order byte of this field is the LU number. The LU number can be any value from 0 to 32. If you code 0, SNA chooses the first available LU; otherwise, SNA uses the value you coded. In either case the PU number and LU number are returned in the second code word of the TCB ($TCBCO2), with the PU number in the high-order byte. For example:

LU	Action
0004	The SNA instruction uses LU #4 on PU #1 and returns 0104 in $TCBCO2. All other requests must have LU equal to 0104.
0104	The SNA instruction uses LU #4 on PU #1 and returns 0104 in $TCBCO2.
0220	The SNA instruction uses LU #32 on PU #2 and returns 0220 in $TCBCO2.
0000	SNA chooses LU #1 on PU #1 for the application and returns 0101 in $TCBCO2. All other requests must have LU equal to 0101. (This example assumes LU #1 is the first available LU.)
0400	SNA chooses LU #8 on PU #4 and returns 0408 in $TCBCO2. All other requests must have LU equal to 0408. (This example assumes LU #8 is the first available LU.)

Do not code this parameter if you code the HOLDLU parameter on the NETINIT request; otherwise, you must code LU. LU must be coded if PTHRU = YES.

HOLDLU (required if not using LU)
The label of a 1-word field that specifies the session LU number to be reestablished after receiving UNBIND HOLD. Code the high-order byte with the PU number and the low-order byte with the LU number. The PU number can be any value from 0 to 4, with 0 implying PU #1. The LU number is the number of the LU to use for the request (on the specified PU) and can be any value from 1 to 32.

Do not specify this parameter if you code the LU parameter on this NETINIT request; otherwise, you must code HOLDLU. HOLDLU is invalid if PTHRU = YES.

HOSTID (required)
The label of the SNA NETHOST data definition. You must specify this parameter.

ACQUIRE
Specifies whether the host or SNA support initiates the session:

YES (default) SNA initiates the session on behalf of the application program.

NO The host initiates the session.

SNA ignores ACQUIRE if PTHRU = YES.

ATTNEV
The address of an event control block (ECB) to be posted when an attention event occurs while no SNA operations are active. You should issue a NETGET instruction to determine whether the event is for status information or data if the session is not a passthru session. If the session is a passthru session (PTHRU = YES), the post code in the first word of the ECB indicates the new status of the passthru session. ATTNEV is recommended if PTHRU = YES..

ERRCODE
The label of a 4-byte data area where SNA stores extended error information. If specified and the SNA operation returns a negative return code (other than −1), this data field identifies the SNA instruction and the related SNA or EDX function that failed plus the return code of the SNA or EDX function. The ERRCODE parameter is not supported for duplex sessions. The following data is included:

Byte 1 The SNA operation in progress when it encountered the error:

00 - NETINIT
01 - NETPUT
02 - NETGET
03 - NETCTL
05 - NETTERM.

Byte 2 The Event Driven Executive or SNA base function that reported the error. The following hexadecimal codes return:

01 - NETOPEN
02 - NETRECV
03 - NETSEND
04 - NETCLOSE
05 - NETBIND
06 - NETUBND
08 - BIND event post code
0A - READ
0B - WRITE
0C - Session termination.

Note: For additional information on the return codes for these functions, refer to Chapter 11, "Extended Error Information."

Bytes 3 and 4 The error return code from the Event Driven Executive or SNA base function.

ERRCODE is a valid option if PTHRU=YES.

EXIT

The label of the Series/1 application error processing routine. Control passes to this label if a negative return code *other than* −1 returns to your application.

FULLDPX

Whether the transmission mode for the session to be established is duplex or half-duplex. Specify one of the following:

NO (default) Establish the session as half-duplex.

YES Establish the session as duplex.

Note: If you code FULLDPX=YES, you cannot use message resynchronization and attention event processing.

SNA ignores FULLDPX if PTHRU=YES.

LUSWAIT

Specifies whether NETINIT should wait for activation of the LU-SSCP session:

YES (default) Forces NETINIT to wait for activation

NO NETINIT fails if the LU-SSCP session is not active

If PTHRU=YES, then you must specify LUSWAIT=YES.

MSGDATA
The label of a 6-byte data area where the SNA support stores information about messages exchanged during the session.

If RESYNC=YES or INIT, the following considerations apply:

- If RTYPE=DISK, SNA support ignores MSGDATA. RDSCB is a required parameter, if you specify RTYPE=DISK.
- If RTYPE=STG, MSGDATA is required. SNA support uses the MSGDATA area specified for resynchronization data. SNA support returns the resynchronization data upon successful completion of an SNA operation. The resynchronization data is reserved for SNA use only, and you must supply it on the NETINIT instruction when restarting the session.

If RESYNC=NO, MSGDATA is optional. When you specify MSGDATA, SNA uses the area to hold message data. When a NETPUT LAST=YES operation succeeds, SNA stores the number assigned to the host in the first and second bytes of the data area. The remaining bytes of the area are reserved for SNA use only.

MSGDATA is invalid if PTHRU=YES.

MSGPRIO
Sets the priority of all outbound messages for the LU while in session. The value overrides the value specified on the SNALU statement. When the session ends, the value reverts to the value on the SNALU statement. Use this parameter to specify which LU is to have its messages serviced first by SDLC if a bottleneck occurs. You must code a label of a value from 1 to 255 (1 is the highest priority). The default is MSGPRIO=255. See "Prioritizing Messages" on page 2-21 or "Prioritizing Messages" on page 3-19.

> **Warning:** If message prioritization is *not* important to an LU, then use the default value of 255 so as not to degrade overall performance.

MSGPRIO is a valid option if PTHRU=YES.

PACB
May specify if PTHRU=YES; invalid if PTHRU=NO. PACB establishes the logical connection between two remotely connected applications. The primary side of the passthru session establishes communications with the remote secondary application using NETOPEN, while the secondary side of the passthru establishes communications with the remote primary application using NETINIT. You must issue NETINIT and NETOPEN for a passthru session to exist. The second command (either NETOPEN or NETINIT) must specify PACB to link the two commands. You *must not* specify PACB on the first command issued for the passthru session. For NETINIT, PACB contains the session access control block (ACB) returned to the application upon issuance of NETOPEN.

PTHRU
Specifies whether the session is a passthru session or not. For passthru, you must specify PTHRU=YES. PTHRU=NO is the default; it specifies that the session is not a passthru and that the application issuing the NETINIT is the application communicating with the host application. If the session is a passthru session, then attention event (ATTNEV) usage is highly recommended.

Px

Parameter naming operands. See "Using the Parameter-Naming Operands" on page 1-11 for a further description. For NETINIT, Px corresponds to the following parameters:

P1 - LU/HOLDLU
P2 - MSGDATA
P3 - SESSPRM
P4 - ATTNEV
P5 - RDSCB/PACB
P6 - ERRCODE
P7 - MSGPRIO.

Note: In coding your program, you can (if resources on your system are available) establish multiple sessions per task. All tasks using these sessions must be within the same program.

RDSCB

The address of an opened data set control block (DSCB) to be used by SNA resynchronization processing. You must specify this parameter if you specify RTYPE=DISK; otherwise, SNA support ignores the RDSCB parameter.

RDSCB is invalid if PTHRU=YES.

RESYNC

Whether you want session resynchronization support:

YES (default) You want session resynchronization support. Use the contents of the resynchronization data set during session establishment.

INIT You want session resynchronization support. Initialize the content of the resynchronization data set during session establishment.

NO You do not desire session resynchronization support.

You must specify RESYNC=NO if PTHRU=YES or if FULLDPX=YES.

RTYPE

Whether you save session resynchronization data on disk or in storage. SNA support ignores this parameter if you specify RESYNC=NO. Code one of the following:

DISK (default) Save session resynchronization data on disk. If you specify DISK, you must code the RDSCB parameter.

Note: Your program must open and close the resynchronization data set.

STG Save session resynchronization data in storage. If you specify STG, you must code the MSGDATA parameter.

SNA ignores this parameter if you code RESYNC=NO.

RTYPE is invalid if PTHRU=YES.

SESSPRM

The label of the data area where SNA support stores the session establishment parameters (BIND) when received from the host. Upon successful completion of the NETINIT operation, the area contains the parameters. The size of this area must be 256 bytes.

SESSPRM is invalid if PTHRU=YES.

Using the NETINIT Instruction

This section describes the operation of the NETINIT instruction when used in your application program. Included in this discussion are:

- Summary of SNA protocols for NETINIT
- Attention event
- Message resynchronization and recovery
- Message numbers
- Resynchronization data set
- Message transmission modes
- Changing a secondary LU's session partner
- Message passthru.

Summary of SNA Protocols for NETINIT

Figure 5-1 lists the SNA command protocols associated with NETINIT parameters.

SNA Parameters	SNA Protocols
ACQUIRE=YES	SNA INIT-SELF command sent to host.
ACQUIRE=NO	SNA waits for the host to send a BIND command for the session.
RESYNC=INIT	If the SNA STSN command is received with sequence numbers, the response is sent using the same sequence numbers. For an established session: • NETPUT does not complete until a reply is received from the host. • All FMD chains go to the host with the CD indicator set.
RESYNC=YES	If the SNA STSN command received is to test the sequence numbers, the response goes with the sequence numbers from the resynchronization data set. For an established session: • NETPUT does not complete until it receives a reply from the host. • All FMD chains go to the host with the CD indicator set.

Figure 5-1 (Part 1 of 2). SNA Protocols for NETINIT

SNA Parameters	SNA Protocols
RESYNC=NO	If an SNA STSN command to test the sequence numbers is received, the response indicates that sequence numbers are not kept. For an established session, all FMD chains go to the host with the CD indicator setting determined by the INVITE parameter of the NETPUT instruction.

Figure 5-1 (Part 2 of 2). SNA Protocols for NETINIT

Attention Event

Series/1 SNA support posts the attention event (ATTNEV parameter) specified for the session when a message arrives from the host. SNA will also post the attention event specified for the session when the Series/1 operator attempts to deactivate $NETx by entering operator commands SNADACT or PUDACT. When SNA posts the attention event, you should issue a NETGET instruction on the session to receive the message arriving from the host or the status information if the session is not passthru.

For passthru support (PTHRU=YES), SNA posts the attention event whenever status is available. The post code gives the most current status of the passthru session. It is highly recommended that you use the attention event when PTHRU=YES.

Note: It is possible for you to post the attention event for a message that is not passed to you by SNA. In this case, the NETGET operation fails with a "no messages available" return code.

SNA support does not support attention event processing on a session using duplex transmission mode (FULLDPX=YES).

Message Recovery and Resynchronization

Recovery and resynchronization ensures integrity of message flow after a session restart. The SNA application program can select message resynchronization support for a session with a session establishment (NETINIT RESYNC=YES or INIT) operation parameter.

Message recovery may be necessary because of the restart of the host subsystem or restart of a failed session. The host uses an SNA command to resynchronize message flow with Series/1 SNA support. This support handles the resynchronization protocol.

When you select the support, an operation to send a message to the host does not complete until it receives a reply indicating or implying successful receipt of the message. Also, the operation grants the right-to-send to the host on every message sent.

When you code RESYNC=YES, the return code to NETINIT indicates when you may need to start recovery procedures. You must write the SNA application program to meet the requirements of the host subsystem for maintaining recoverability. When the NETINIT return code indicates that a message was lost,

the SNA application program should initiate the recovery procedure required by the host application program.

When you code RESYNC=INIT, the session restarts at the point of message exchange last recorded by the host. The host then proceeds without trying to recover any lost messages. You are not notified of any lost messages. RESYNC=INIT also initializes the contents of the resynchronization data for the session.

Notes:

1. SNA support does not support message recovery and resynchronization processing on a session using duplex transmission mode (FULLDPX=YES).
2. SNA does not support message recovery and resynchronization if PTHRU=YES.

Message Numbers

During a session, the messages exchanged between the host subsystem and the SNA application are numbered. The host assigns numbers to inbound messages; Series/1 SNA support assigns numbers to outbound messages.

When you establish the session with resynchronization support (RESYNC=YES or INIT), SNA support saves the message number of the last message flowing in each direction for the resynchronization process.

For sessions without resynchronization (RESYNC=NO), the message numbers optionally return to the Series/1 SNA application program on send (NETPUT) operations. You can use these numbers to correlate negative responses from the host to messages transmitted.

If RESYNC=YES or INIT, there is no need for you to obtain the message number of transmitted messages, as only one message may be pending verification by the host at any time.

Resynchronization Data Set

For sessions established with resynchronization support, Series/1 SNA support maintains the resynchronization data necessary for recovery/resynchronization. You can maintain the resynchronization data for a session in an EDX disk or diskette data set or in Series/1 main storage. The resynchronization data is not used unless you establish the session with resynchronization support.

When you specify disk/diskette storage (RTYPE=DISK), Series/1 SNA support saves the message resynchronization data in the data set specified on the NETINIT RDSCB parameter. You should allocate the resynchronization data set with a length of 256 bytes. The Series/1 application opens and closes the resynchronization data set.

When you specify main storage (RTYPE=STG), SNA support maintains the resynchronization data in an area specified by the NETINIT MSGDATA parameter.

For more information on the resynchronization data set, see Chapter 10, "Message Recovery and Resynchronization."

Message Transmission Modes

The remote session partners determine the mode for passthru.

Series/1 SNA transmits messages between session partners in one of two modes: duplex or half-duplex.

The primary LU defines whether message transmission is to be half-duplex or duplex when SNA establishes a session with a secondary LU. You can specify the mode of message transmission for the Series/1 using the NETINIT instruction.

Message Transmission in Half-Duplex Mode

Half-duplex message transmission means that session partners must take turns sending messages to the other session partner. To send a message, a session partner must have the *right-to-send.* The session partner that has the right-to-send may grant it to its session partner. The session partner starting the transaction begins with the right-to-send.

When no transaction is active during the session, both session partners have the right-to-send. If both partners send at the same time, the Series/1 transaction starts and rejects the host application transaction. Later, the host can retry the rejected transaction.

Message Transmission in Duplex Mode

Duplex message transfer allows messages (including function management data and normal data flow control commands) to be sent and received at the same time.

A send operation may be awaiting completion (such as a response from the host) at the same time a receive operation is being processed in the same session. Each session partner always has the right-to-send at any time during a session.

The Series/1 SNA application programmer is able to emulate the Customer Information Control System (CICS) 3650 Pipeline Logical Unit when using duplex. The Series/1 SNA brackets protocol is compatible with that used for the Pipeline Logical Unit.

Note: Series/1 SNA manages duplex and half-duplex message transmission before messages are transmitted over the data link. Message transmission over the data link is always half-duplex using SDLC communications protocol.

The following considerations and restrictions apply to Series/1 SNA duplex message transfer:

- Only two tasks can be in session under the same LU at one time. For example, one task can issue NETPUT instructions and the second task can issue NETGET instructions.
- The NETTERM instruction is the only other operation allowed on a different task.
- Only single element chains are allowed. Your application must ensure that all data sent via NETPUT (whether it is sent with one NETPUT LAST=YES or a series of NETPUT LAST=NO operations) fits within one request unit (RU). A negative return code (system error) returns to the application if the data length for any one message exceeds the RU length.

- Message resynchronization is not supported for a duplex session. SNA ignores all parameters on NETINIT dealing with resynchronization support.

  Since Series/1 SNA does not support resynchronization, it sends a positive response to the host for each definite response request (RQD) received without SNA protocol errors. The response goes to the host once the entire message transfers to the application program. The application program cannot send any responses to the host (that is, NETCTL TYPE=ACCEPT/REJECT is not allowed with duplex transmission).

- Series/1 SNA completely handles bracket protocol. It ignores the EOT parameter on NETPUT. If the BIND parameters specify that brackets are allowed, Series/1 SNA always sets both the begin bracket and end bracket indicators on each request being sent. Otherwise, it sends no bracket indicators.

- The change direction indicator has no meaning in duplex. Therefore, SNA ignores the INVITE parameter on NETPUT. It also ignores change direction indicators received from the host and does not report "right to send received" return codes on NETGET or NETCTL.

- Function management header support is the same as with half-duplex.

The following restrictions apply to simultaneous operations:

- You cannot initiate an operation on a session until the NETINIT operation for that session completes. This is consistent with half-duplex operation.

- The ERRCODE parameter is not supported for duplex sessions.

- Any operations which have not completed when SNA issues a NETTERM completes with a "NETTERM in progress" (−15) return code. Any operations issued in a session after NETTERM was issued but not yet completed, completes with an "operation in progress" return code. This is consistent with half-duplex operation.

- No operation may overlap with another identical operation; for example, there can be only one NETPUT in progress at any one point in time.

- You can issue NETGET concurrently with an outstanding NETPUT or NETCTL.

- You can issue NETPUT or NETCTL concurrently with an outstanding NETGET.

- NETPUT and NETCTL TYPE=QEC|RELQ|SIG|LUSTAT|RTR cannot overlap. The SNA architecture indicates that only one definite response request can go to the host at any given time.

- You must issue NETCTL TYPE=RECV synchronously. This restriction allows NETCTL TYPE=RECV operations to overlap with NETPUT, NETGET, and other NETCTL operations.

How to Design an Application Using Duplex

One way to design an application using duplex transmission is to have one task that is a PUT task; that is, a task receiving input from a keyboard type of device, and issuing NETPUTs to SNA when operator input is received. Another task is the GET task; that is, a task receiving input from the host via SNA and sending the output to some type of display device. These two tasks run simultaneously, with little cross task communications necessary.

Design the GET task to issue a NETGET and wait for data to be received from the host. Once your application receives data, it sends information to the display device and reissues the NETGET. Design the PUT task to wait for information from the keyboard, issue a NETPUT to send the message to the host, then wait again for information from the keyboard.

NETGETs issued under duplex transmission mode cannot use the attention event. Since SNA allows concurrent NETPUT and NETCTL operations, control does not return to your application after a NETGET until it receives data.

Note: Duplex processing implies that all NETGET operations are suspended until a request is received from the host. The "no data available" (−26) return code does not return on a duplex session.

As with half-duplex, many errors or unexpected results that are detected during NETPUT and NETGET operations result in a "status available" (−17) return code. You must follow this return code with a NETCTL TYPE=RECV operation, in order to receive the status involved. When status is available, any NETPUT, NETGET, or other NETCTL operations results in the "status available" return code.

Status information returned on the NETCTL TYPE=RECV returns to the application in the order received on the session. Correlation of this status information to prior operations is the application's responsibility.

Changing a Secondary Logical Unit's Session Partner

A primary LU (PLU) can change its session with a secondary LU (SLU) so that the SLU communicates with another application executing on the host system. The new application is also a primary LU in the SNA network. This is accomplished when the PLU sends a SNA UNBIND HOLD command to the Series/1. You can then specify NETTERM HOLD=YES followed by NETINIT HOLDLU= to establish a session with the new PLU.

For example, different applications executing on the host system can communicate with the same application executing at the SLU. A primary LU accomplishes this change in communication by putting the secondary in a holding status and by transferring its session to another PLU at the host system. The session is reestablished with the secondary LU without disrupting the secondary LU's operation in the SNA network.

The host can issue an UNBIND HOLD during a session between the primary LU and a secondary logical unit. The "hold" feature of UNBIND allows the transfer of a session from one PLU to another PLU. You must specify the HOLDLU parameter of the NETINIT instruction for the session to be reestablished from the host system. You must have previously issued a successful NETTERM HOLD=YES operation for a session after receiving the return code (−20) telling you that UNBIND HOLD was received.

Receiving Additional Information About the Session

A number of parameters available on the NETINIT instruction, if coded, give you valuable data about your session at startup, while operating, and at session deactivation. You can use these fields for further verification of certain session protocols, for data flow control during a session, or for problem determination after a session terminates.

If your application needs to further investigate the session rules provided in the BIND command for the session, you should code the SESSPRM parameter to specify an area where SNA can copy these parameters after receiving the BIND. Requesting a copy of the BIND command helps you perform further BIND parameter checking than SNA provides by basic BIND checking. This can help determine if the application can support the session rules as defined by the host. Another reason for requesting a copy of the BIND command is to investigate the rules applicable for this session and to use these rules to make data flow control decisions while the session is active.

Even if RESYNC=NO, you can determine what messages were sent to the host subsystem. Applications that are highly concerned about data integrity within the SNA session can use this knowledge. For RESYNC=YES or INIT, SNA keeps track of messages and reports missed or lost messages. However, if RESYNC=NO, you can use the MSGDATA parameter to give an application some clue as to what messages were successfully sent to the host. SNA stores the SNA sequence number of the last complete message (LAST=YES on NETPUT) successfully sent to the host in this area, and the application can use this to follow the record of data transfer for the session.

Finally, you can greatly help in problem determination of the cause for a session failure by using the ERRCODE parameter. Use of this parameter is strongly recommended because of the aid it gives in problem determination after a session abnormally terminates (−16) or a system error occurs (−14). In either case, SNA stores additional error information in this field to help further determine the cause of the failure. Without this parameter, finding the exact cause of a failure may be impossible.

Waiting for the LU-SSCP Session

SNA support automatically activates the LU-SSCP session when the host system requests activation of the LU. No LU-LU data flow can occur for a particular LU until the LU-SSCP session is active. If SNA receives a NETINIT request for a LU whose LU-SSCP session is not active, it usually waits until the session activates and then tries to complete the request. You may want to know immediately if the LU-SSCP session is down for a LU, because your application cannot afford to wait for the session to activate if it is not already active (real-time processing); or you may want to know immediately so you can tell your network operator about the problem. In these cases, using LUSWAIT=NO specifies that if the LU-SSCP session is not active, immediately cause the request to fail and report the error to the application.

Prioritizing Outbound Messages for a Session

You can prioritize outbound messages on a session basis by LU so that messages from one session have priority over messages from another session. This support is the same as the message priority support provided while defining the LU (see the MSGPRIO parameter under "SNALU Configuration Statement" on page 2-19), except that the application can override the generated priority for the length of the session. Some reasons for using this support are:

- An application may want to ensure a particular priority in relationship to other applications, even though the application may use different LUs at different times.
- An application may request a session specifying that SNA can choose any LU. Therefore, each time the application starts, it keeps the same priority, no matter which LU SNA chooses for the session.

The priority given by the MSGPRIO parameter on NETINIT remains in effect for that particular LU until the session terminates. After the session terminates, SNA once again uses the priority specified in the PU definition (MSGPRIO on the SNALU command) for all outbound messages. Message priority applies only to outbound messages. See "Prioritizing Messages" on page 2-21 and "Prioritizing Messages" on page 3-19 for more details.

> **Warning:** Message prioritization can affect overall performance of the SNA network. Use message prioritization only where required.

Message Passthru Support

Message passthru passes messages from a remote LU, through the Series/1, to another remote LU. (The Series/1 is transparent to the remote LUs.) Message passthru is very useful if some remote secondary LUs need access to a critical application in the host or vice versa. While the host is up, SNA uses message passthru to establish a logical link through the Series/1, connecting the two remote LUs in session, and monitors the session. If the host becomes unavailable and the session terminates between the remote LUs, passthru detects the error, and the session can pass to the Series/1 until the host becomes available. If the host is down, the remote secondary LU still has access to at least some functions provided by the Series/1 until the host resumes operation.

A passthru session is the logical connection between two local LUs defined in the Series/1, which are, in turn, attached logically to two remote LUs. The goal is to establish the passthru session so that the two remote LUs can establish an actual LU-LU session between themselves. The passthru session consists of primary and secondary sides. The primary side refers to the connection between a local primary LU on the Series/1 to a remote secondary LU, established using the Primary SNA NETOPEN command. The secondary side refers to the connection between a local secondary LU on the Series/1 to a remote primary LU, established using the SNA NETINIT command. The PACB parameter logically links the primary side and secondary side.

While passthru is active, all messages pass through the Series/1 without any application intervention. You cannot use NETPUT, NETGET, or NETCTL at any time on a passthru session. NETTERM terminates the passthru session, which terminates the actual LU-LU session if that session is active. SNA notifies the application of any major change in the passthru session or LU-LU session (session is active, primary side terminates, session terminates) by posting the BIND event specified on the NETOPEN command and the ATTNEV event specified on the NETINIT command. The passthru application can then take appropriate action.

You must have Primary SNA for passthru sessions. For more information on passthru support, refer to *Primary Systems Network Architecture Programming Guide*.

NETINIT Coding Illustrations

The coding illustrations presented here show variations of the use of the NETINIT instruction to establish a session.

Session with Resynchronization Data to Disk

This coding illustration shows a session being established where the resynchronization data resides on disk; in addition:

- Series/1 SNA initiates the session with the host. SNA saves the extended information at location SAVERC.
- The resynchronization data set RDSCB is RESTART.
- The LU is number 1, on PU 1, specified at location NETLU.
- NETINIT waits if the LU-SSCP session is not active.
- Priority for all outbound messages is 40 while the LU is in session.

```
      NETINIT  LU=NETLU,                                          C
               HOSTID=SNAID,                                      C
               ACQUIRE=YES,                                       C
               ERRCODE=SAVERC,                                    C
               RESYNC=YES,                                        C
               RTYPE=DISK,                                        C
               RDSCB=RESTART,                                     C
               LUSWAIT=YES,                                       C
               MSGPRIO=MPRIO
            •
            •
            •
NETLU     DATA  F'1'
SAVERC    DATA  2F'0'
RESTART   DSCB  DS#=RSYNC,DSNAME=RSYNDSCB
SNAID     NETHOST   ISAPPID=IMS,ISMODE=INQUIRY
MPRIO     DATA  F'40'
```

Session with Resynchronization Data to Storage

This coding illustration shows a session being established with resynchronization where the resynchronization data resides in storage. In addition:

- Series/1 SNA support waits for session initiation by the host.
- SNA initializes the resynchronization data when the session starts.
- SNA saves the resynchronization data at address RDATA.

```
     NETINIT  LU=NETLU,                                          C
              HOSTID=SNAID,                                      C
              ACQUIRE=NO,                                        C
              LUSWAIT=YES,                                       C
              RESYNC=INIT,                                       C
              RTYPE=STG,                                         C
              MSGDATA=RDATA
           •
           •
           •
NETLU    DATA  F'1'
RDATA    DATA  3F'0'
SNAID    NETHOST    ISAPPID=CICS,ISMODE=INQUIRY
```

Session without Resynchronization

This coding illustration establishes a session without resynchronization support. SNA saves the message numbers at address MDATA.

```
   NETINIT  LU=NETLU,          ANY LU ON PU 3                    C
            HOSTID=SNAID,      SPECIFY WHICH HOST APPL           C
            ACQUIRE=NO,        LET THE HOST START THE SESSION    C
            RESYNC=NO,         NO MESSAGE RESYNCHRONIZATION      C
            MSGDATA=MDATA,     PASS BACK THE SEQUENCE NUMBERS    C
            LUSWAIT=NO,        FAIL IF LU-SSCP SESSION IS INACTIVE C
            MSGPRIO=MPRIO      MESSAGE PRIORITY IS 80
         •
         •
         •
NETLU    DATA  X'0300'            PU NUMBER 3, ANY LU
MDATA    DATA  3F'0'              MSGDATA AREA
SNAID    NETHOST ISAPPID=JES2,ISMODE=RMT26  TALK TO JES2 APPL ON HOST
MPRIO    DATA  F'80'               MESSAGE PRIORITY = 80
```

NETINIT Return Codes

NETINIT return codes go in the first word of the task control block ($TCBCO) of the task issuing the instruction.

If you code the ERRCODE operand on the NETINIT instruction, additional error information returns, when appropriate, to the area you specify.

The positive return codes from NETINIT contain bit-significant values to allow for efficient analysis in the Series/1 SNA application. For a description of the bit-significant values, see "NETINIT Return Codes for STSN Processing" on page 10-21.

The following are the decimal return codes that can return from a NETINIT operation:

Return Code	Condition
81	Message flow to host cold-started, message to host possibly lost. Message flow from host cold-started, no messages from host lost.
49	Message flow to host cold-started, message to host lost. Message flow from host cold-started no messages from host lost.
32	Message to host lost.
19	Message flow to host cold-started. Message flow from host cold-started, message from host lost.
17	Message flow to host cold-started, no messages to host lost. Message flow from host cold-started, no messages from host lost.
4	Partially presented message from host lost.
2	Unpresented message from host lost.
-1	Operation successful.
-7	SNA is in the process of loading or unloading and is temporarily not usable.
-8	SNA deactivating.
-12	Invalid LU number.
-14	SNA system error.
-15	NETTERM in progress.
-16	Session abnormally terminated by host.
-19	$SNA was never loaded or $SNA was unloaded.
-23	Invalid PU number (must be from 0 to 4).
-24	The selected PU is not active.
-26	LU already open.
-27	No LU available.
-30	BIND from host rejected.
-31	STSN error.
-32	No NETTERM HOLD=YES issued.
-33	Invalid priority coded. Must be a number from 1 through 255.

Attention Event Post Codes for Passthru

Post Code	Description
1	UNBIND sent by the remote primary LU to the remote secondary LU.
-1	BIND sent by the remote primary LU to the remote secondary LU.
-9	Passthru session terminated; issue NETTERM.
-14	No stack available to send message through the passthru link— the primary SNA PU in the Series/1 may be deactivating; issue NETTERM.

Chapter 6. Sending Messages - NETPUT

To transmit messages from a Series/1 application program to the host application program, use the NETPUT instruction. You can issue NETPUT only after a nonpassthru session is successfully established. You can send a complete message to the host in one NETPUT operation. Or, if necessary, you can send a complete message to the host by multiple NETPUT operations. See Figure 6-2 on page 6-4

If no transaction is active on the session when you issue NETPUT, Series/1 SNA support starts a transaction with the message. In the event that the host application program also attempts to start a transaction, SNA rejects the host transaction attempt. A return code sent when the Series/1 NETPUT operation completes reflects the rejection of the host transaction. When the Series/1 application program transaction completes, the host application can retry the rejected transaction.

You must have the right-to-send for NETPUT to complete successfully. If you are receiving and need to send, issue the NETCTL instruction and specify TYPE=SIG to request the right-to-send. When no transaction is active on the session, both you and the host have the right-to-send.

You can cancel a message being sent by the Series/1 at any time prior to its completion by issuing a NETCTL instruction and specifying TYPE=CANCEL. The host discards any part of the message it has already received.

The host can request Series/1 SNA support to suspend sending messages temporarily. When this occurs and you issue a NETPUT instruction, the operation fails with a "message flow suspended" return code. The host can release your suspended state at a later time. Unless you issue a NETGET instruction, you retain the right-to-send while suspended. See Chapter 8, "Controlling Message Exchange - NETCTL" for more information.

NETPUT is invalid for passthru sessions.

The following are the coding syntax and parameter descriptions of the NETPUT instruction.

```
label        NETPUT     LU=,BUFF=,BYTES=,EOT=,EXIT=,                    C
                        FMH=,INVITE=,LAST=,P1=,P2=,P3=,                  C
                        VERIFY=

Required:    LU,BUFF,BYTES
Defaults:    EOT=NO,FMH=NO,INVITE=YES,
             LAST=YES,VERIFY=NO
Indexable:   none
```

LU (required)
The label of a 1-word field specifying the LU and PU numbers for the session. Code the high-order byte with the PU number and the low-order byte with the LU number. The PU number can be any value from 0 to 4, with 0 implying PU #1. The LU number is the number of the LU to use for the request (on the specified PU) and can be any value from 1 to 32.

BUFF (required)
The address of a field where the message, or partial message, to be sent is stored. You must specify this parameter.

BYTES (required)
The label of a 1-word field containing the number of bytes in the message, or partial message to be sent. You must specify this parameter.

EOT
Whether the transaction should end with the message. SNA ignores this parameter if coded on other than the first NETPUT issued for the message. Code one of the following:

YES	End the transaction.
NO (default)	Do not end the transaction.

EXIT
The label of the Series/1 application error processing routine. Control passes to this label if any *negative* return code *less than* −1 returns to your application.

FMH
Whether the message contains function management (FM) headers. SNA ignores this parameter if coded on other than the first NETPUT issued for the message. Code one of the following:

YES	The message contains FM headers.
NO (default)	The message contains no FM headers.

INVITE
Whether to grant the right-to-send to the host with this message. SNA ignores this parameter unless you specify LAST=YES. Code one of the following:

YES (default)	Grant the right-to-send.
NO	Do not grant the right-to-send.

LAST
Whether this is the last NETPUT operation for the message. Code one of the following:

YES (default)	This is the last NETPUT operation for the message.
NO	This is not the last NETPUT operation for the message.

Px
Parameter naming operands. See "Using the Parameter-Naming Operands" on page 1-11 for a further description. For NETPUT, Px corresponds to the following parameters:

P1 LU

P2 BUFF

P3 BYTES

VERIFY
Whether to verify receipt of this message by the host. SNA ignores this parameter unless you specify LAST=YES. Code one of the following:

YES	Verify receipt of the message.
NO (default)	Do not verify receipt of the message.

Using the NETPUT Instruction

This section describes the operation of the NETPUT instruction in your application program. Included in this discussion is:

- Summary of protocols for NETPUT
- End-of-transaction
- Function Management Headers
- Right-to-send
- Message acknowledgment.

Summary of SNA Protocols for NETPUT

Figure 6-1 lists the SNA command protocols associated with NETPUT parameters.

SNA Parameters	SNA Protocols
INVITE = YES	Chain sent to host with CD indicator set (if EOT = YES, CD is not sent but is implied)
VERIFY = YES	Chain sent to host as RQD1. NETPUT does not complete until a response comes back from the host.
EOT = YES (except for CICS)	Chain sent to host with EB indicator set. **Note:** CICS may not permit this parameter. Application may use INVITE = YES.
LAST = YES	Chain ended (EC indicator set) when data specified goes to the host.
FMH = YES	Chain sent to the host with FI indicator set.

Figure 6-1. SNA Protocols for NETPUT

End-of-Transaction

For an illustration of end-of-message and end-of-transaction coding see Figure 6-2 on page 6-4. The NETPUT operations constitute a 4-message transaction. The first three NETPUT operations result in three separate messages; the LAST = YES parameter in each message makes this qualification. The following three NETPUT operations result in the fourth message. Note that EOT = YES appears on the first part of the message while the continuation of the message appears by the LAST = NO parameter on parts 1 and 2 of the message. The message (as well as the transaction) concludes on the third part of the message (LAST = YES).

NETPUT ...,LAST=YES	(message 1)
NETPUT ...,LAST=YES	(message 2)
NETPUT ...,LAST=YES	(message 3)
NETPUT ...,EOT=YES,LAST=NO	(message 4, part 1)
NETPUT ...,LAST=NO	(message 4, part 2)
NETPUT ...,LAST=YES	(message 4, part 3)

Figure 6-2. NETPUT End-of-Message and End-of-Transaction Coding

Function Management Headers

You can supply function management headers (FMH) as data to be sent with the rest of the data in the message. You must build the function management headers and include them as required in the data being sent to the host program. Coding FMH=YES indicates to the receiving host application that the message has a prefixed FMH. Series/1 SNA support does not check function management headers for correct format or validity.

Right-to-Send

The INVITE parameter controls whether the Series/1 application program gives the right-to-send to the host application program. INVITE is valid only when used with the last NETPUT (LAST=YES) of a message. INVITE=YES grants the right-to-send to the host application program. INVITE=NO indicates that the Series/1 application program retains the right-to-send.

Note: NETINIT RESYNC=YES and NETINIT RESYNC=INIT sessions send all messages as if you coded INVITE=YES.

Message Acknowledgment

When you specify VERIFY=YES, the NETPUT operation does not complete until the host acknowledges successful receipt of the message. The acknowledgment can be a positive or a negative acknowledgment. A positive acknowledgment indicates successful receipt of an acceptable message. The host subsystem issues a negative acknowledgment when it finds an error in the message from the Series/1 application or when an unusual condition in the host prevents the subsystem from accepting the message.

Note: For NETINIT RESYNC=YES or NETINIT RESYNC=INIT sessions, NETPUT operations do not complete until the host acknowledges successful receipt of the message.

Sometimes when you specify VERIFY=YES, the host application sends a SIGNAL command, requesting the right-to-send, before Series/1 SNA support can receive a response to the data. When this occurs, the NETPUT command does not complete until SNA support receives a response to the NETPUT data. The next SNA command you issue completes with a "status available" (−17) return code. The NETCTL TYPE=RECV indicates that SNA support received the signal command.

When you specify VERIFY=NO (the default), the NETPUT operation completes when the message goes to the host. In this case, the operation does not wait for an acknowledgment of successful receipt of the message.

Receipt of an acknowledgment to any message from the host implies positive acknowledgment of all messages sent up to that point. Depending on the host application, you can send several messages with VERIFY=NO and then send one with VERIFY=YES to determine that all the previous messages were acceptable.

The host may send a negative acknowledgment to any message sent since the last acknowledged message. A negative acknowledgment carries sense data describing the type of the error condition detected. It also carries the message identification of the message being negatively acknowledged.

When the host sends a negative acknowledgment to a message sent with VERIFY=NO, it arrives after NETPUT completes. In this case, a subsequent Series/1 SNA operation on the same session fails with a "status available" return code.

When you receive a "status available" (−17) return code, it means the current operation failed, and a NETCTL TYPE=RECV operation must be the next SNA operation you issue on the session. The return code from NETCTL TYPE=RECV informs you that you received a negative acknowledgement. NETCTL TYPE=RECV also returns to you the sense data from the negative acknowledgment and the message identification of the message in error. The meaning of the sense data depends on the host application. This information may be useful to you in determining the source of the error.

After you determine that you received a negative acknowledgment, you should issue a NETGET operation to receive a message from the host subsystem. This is required since the host has error recovery responsibility for SNA sessions.

NETPUT Coding Illustrations

The following coding illustrations show variations of the use of the NETPUT instruction to send messages.

Sending a Message with a Single NETPUT

This following coding illustration shows a message to the host using one NETPUT instruction; in addition:

- The LU is number 1 at location NETLU.
- The message to be sent is at address OUTBUFF.
- The length of the message to be sent is at address BYTECNT.
- The data is to be sent as a complete message.
- The right-to-send is granted to the host.
- The data includes function management headers.

```
      NETPUT   LU=NETLU,                                          C
               BUFF=OUTBUFF,                                      C
               BYTES=BYTECNT,                                     C
               INVITE=YES,                                        C
               FMH=YES,                                           C
               LAST=YES
            •
            •
            •
NETLU     DATA  F'1'
OUTBUFF   DATA  CL80'MESSAGE'
BYTECNT   DATA  F'80'
```

Sending a Message with Multiple NETPUTs

The following coding illustration shows one message (in three parts) being sent to the host with multiple NETPUT instructions. In addition:

- The lengths of partial messages to be sent are at addresses BYTECNT1, BYTECNT2, and BYTECNT3.
- The receipt of the message by the host is to be verified.
- The transaction ends with the message.

```
      NETPUT   LU=NETLU,                                          C
               BUFF=OUTBUFF1,                                     C
               BYTES=BYTECNT1,                                    C
               EOT=YES,                                           C
               LAST=NO

      NETPUT   LU=NETLU,                                          C
               BUFF=OUTBUFF2,                                     C
               BYTES=BYTECNT2,                                    C
               LAST=NO

      NETPUT   LU=NETLU,                                          C
               BUFF=OUTBUFF3,                                     C
               BYTES=BYTECNT3,                                    C
               VERIFY=YES,                                        C
               LAST=YES

            .
            .
            .
NETLU    DATA  F'1'
OUTBUFF1 DATA  CL40'MESSAGE PART 1'
OUTBUFF2 DATA  CL20'MESSAGE PART 2'
OUTBUFF3 DATA  CL20'MESSAGE PART 3'
BYTECNT1 DATA  F'40'
BYTECNT2 DATA  F'20'
BYTECNT3 DATA  F'20'
```

NETPUT Return Codes

The positive return codes from NETPUT have bit-significant values to allow for efficient analysis in the Series/1 SNA application. The bit positions have the following meaning:

```
.... .... .... ...1     Host attempted to start a transaction
```

The valid combinations of the bit positions are listed in the following decimal return codes:

Return Code	Condition
1	Host attempted to start transaction.
-1	Operation successful.
-7	SNA is in the process of loading or unloading and is temporarily not usable.
-8	SNA deactivating.
-9	LU is busy with another operation.
-10	Session does not exist.
-11	Program linked to $NETCMD must issue instruction from same program that issued NETINIT.
-12	Invalid LU number.
-13	Invalid request.
-14	SNA system error.
-15	NETTERM in progress.
-16	Session abnormally terminated by host.
-17	Status available.
-18	Session quiesced.
-19	$SNA was never loaded, or $SNA was unloaded.
-20	UNBIND HOLD received.
-21	More than two tasks running under this LU. The limit is two tasks.
-22	Session reset; CLEAR and SDT commands received.
-23	Invalid PU number (must be between 0 and 4).
-24	Selected PU not active.
-25	Not right-to-send.
-26	RTR sent; issue NETGET to receive host transaction.

Chapter 7. Receiving Messages - NETGET

Use NETGET to ensure that the Series/1 application program receives messages from the host application program. Before using NETGET, you must establish a nonpassthru session with the host.

When the application program issues NETGET, Series/1 SNA support passes message data received from the host's application program into a buffer area provided by NETGET. If the buffer area is not large enough to contain the complete message, you can issue additional NETGET instructions. The return code from NETGET indicates the end of the message.

NETGET is invalid for passthru sessions.

The following is the coding syntax and parameter descriptions of the NETGET instruction.

```
label          NETGET    LU=,BUFF=,BYTES=,RECLEN=,                    C
                         EXIT=,P1=,P2=,P3=,P4=

Required:      LU,BUFF,BYTES,RECLEN
Defaults:      none
Indexable:     none
```

LU (required)
The label of a 1-word field specifying the LU and PU numbers for the session. Code the high-order byte with the PU number and the low-order byte with the LU number. The PU number can be any value from 0 to 4, with 0 implying PU #1. The LU number is the number of the LU to use for the request (on the specified PU) and can be any value from 1 to 32.

BUFF (required)
The address of the field where you want to store the message, or partial message, when received. You must specify this parameter.

BYTES (required)
The label of a 1-word field containing the number of bytes in the area specified by BUFF. You must specify this parameter.

RECLEN (required)
The label of a 1-word field in which SNA will place the actual number of bytes received in the area specified by BUFF. You must specify this parameter.

EXIT
The label of the Series/1 application error processing routine. Control passes to this label if any *negative* return code *less than* −1 returns to your application.

Px

Parameter naming operands. See "Using the Parameter-Naming Operands" on page 1-11 for a further description. For NETGET, Px corresponds to the following parameters:

P1	LU
P2	BUFF
P3	BYTES
P4	RECLEN

Using the NETGET Instruction

This section describes the operation of the NETGET instruction when used in your application program. Included in this discussion are:

- Summary of protocols for NETGET
- Message completion criteria
- Host-initiated transactions
- Right-to-send
- Message acknowledgment
- Function management headers.

Summary of SNA Protocols for NETGET

Figure 7-1 lists the Series/1 SNA command protocols associated with NETGET return codes.

SNA Condition	SNA Protocols
End-of-Message	Chain received from host with EC indicator set.
End-of-Transaction	Chain received from host with EB indicator set.
Start of transaction	Chain received from host with BB indicator set
Response requested	Chain received from host with RQD1 or RQD2 indicated.
Right-to-send	Chain received from host with CD indicator set.
Function management header	Chain received from host with FI indicator set

Figure 7-1. SNA Protocols for NETGET

NETGET Completion Criteria

The operation of the NETGET instruction depends on whether you are using attention event support for the session and whether there is a transaction currently active on the session. When:

- There is no active transaction on the session and no status information is available, a NETGET operation terminates with a "no messages available" (−25) return code.
- A transaction is active on the session and you are not using attention event support, a NETGET operation does not complete normally until the application receives a message from the host or an error condition occurs.

Attention Event Support: When you use attention event support and no status information is available and no message is received from the host, a NETGET operation terminates with a "no messages available" (−25) return code. This is true regardless of whether there is an active transaction on the session.

Also, when you use attention events in your application, you are responsible for coding the WAIT and RESET event sequence.

Status from Host: When the host's application program sends status information, it is reflected by NETGET's return code. The NETGET operation terminates and you receive a "status available" (−17) return code. You should then issue a NETCTL instruction specifying TYPE=RECV to determine the type of the host status. You cannot issue any other Series/1 SNA instructions (except NETTERM) until you issue NETCTL TYPE=RECV.

Host-Initiated Transactions

When the host program attempts to begin a transaction on the session, your NETGET operation terminates with a "host initiated transaction" (−26) return code. Note that no data passes to your buffer area at this time. You accept or reject the host-initiated transaction by your next Series/1 SNA operation on the session.

Accepting the Host-Initiated Transaction: To accept the host-initiated transaction, issue a NETGET or a NETCTL TYPE=ACCEPT instruction. The host-initiated transaction begins when you receive the first message of the transaction.

Assuming that you defined the attention event (named LUEVENT) on the NETINIT ATTNEV parameter, you can accept the host-initiated transaction as illustrated in Figure 7-2.

```
WAIT     LUEVENT (wait for host Bid or BB)
RESET    LUEVENT
NETGET   LU=NETLU,BUFF=INBUFF,                              C
         BYTES=INLGTH,RECLEN=COUNT

(process NETGET return code)
  •
  •
  •
IF       return code = -26     (if Bid received)
   NETGET  LU=NETLU,BUFF=INBUFF,BYTES=INLGTH,RECLEN=COUNT

  (Host BID accepted—receive data)
  •
  •
  •
ENDIF
```

Figure 7-2. Coding Illustration - Processing a Host-Initiated Transaction

After receipt of the host bid, you can optionally reject the host transaction by issuing a NETPUT.

If you receive a return code of −26 from the first NETGET, do not issue another WAIT instruction on the attention event. To receive the data, issue a second NETGET.

You can loop on the second NETGET until you receive all messages in a chain ("end of message received" return code) or until you receive an "end of transaction received" return code, depending on the host application.

When NETGET completes with −26 (because of the host sending BID), SNA waits for a message while you issue a second NETGET. If the −26 return code indicates that the host sent a message with the begin bracket (BB) indicator on, SNA delays processing the message until you issue a second NETGET. In both cases, the second NETGET completes with a return code indicating "start of transaction received."

Rejecting the Host-Initiated Transaction: To reject the host initiated transaction, issue a NETPUT or a NETCTL TYPE=REJECT instruction. Issuing NETPUT instruction causes rejection of the host initiated transaction and starts a transaction for your application. Issuing a NETCTL TYPE=REJECT only rejects the host-initiated transaction. A subsequent NETPUT instruction starts a transaction for your application.

Requesting the Host Application to Retry: After rejection of a host initiated transaction and when no transaction is active on the session, you can request the host to retry the previously rejected transaction. This initiates the same sequence of events discussed above; however, you should not reject the host-initiated transaction again. When you are ready for the host to retry the transaction, issue a single NETGET instruction to both request the host to retry the host-initiated transaction and to receive the first message of the transaction.

When using attention event support, you can request the host to retry the host-initiated transaction in the following manner:

1. Issue a NETCTL TYPE=RTR instruction.
2. Wait on the attention event for the session.
3. Issue a NETGET when the attention event is posted.

NETGET does not complete until the host sends the first message of the transaction. When you receive the first message of the host-initiated transaction, the NETGET return code indicates "start of transaction received."

If the host decides not to retry the previously rejected transaction, the NETGET operation terminates with a "no messages available" (−25) return code.

Right-to-Send

Your host must have the right-to-send in order for it to send to your program. Series/1 SNA support provides the host with the right-to-send if it does not already possess it when the application issues NETGET. You must, by use of NETGET, continue to receive messages until the transaction ends or the right-to-send goes to Series/1 SNA support. The return code indicates which.

You can issue a NETCTL instruction specifying TYPE=SIG to request the host to return the right-to-send to the Series/1 application program. You must continue to receive until the host grants the right-to-send to your application program.

Message Acknowledgement

The Series/1 application can accept or reject received messages. The Series/1 application program rejects a received message by issuing NETCTL TYPE=REJECT. You imply acceptance of the message if you do not reject it and issue another Series/1 SNA instruction.

The host's application may want explicit verification of the Series/1 SNA application receipt of a particular message. The NETGET return code indicates this. To verify the receipt of an accepted message, use NETCTL TYPE=ACCEPT. Also, issuing any Series/1 SNA instruction other than NETCTL TYPE=REJECT implies verification of the message.

Message Processing

If NETGET is receiving a message, NETGET does not complete until either the message ends (receipt of the EC indicator) or the application's buffer is filled. If NETGET receives an end-of-chain indicator, NETGET completes with a positive return code indicating (among other things) "end of message received." The message is in the area specified by BUFF, and the length of the received message is in the area specified by RECLEN.

If the application's buffer is filled, NETGET completes with "operation successful" (−1). This indicates that although NETGET received all the data it can handle, there is more data to be received for the message. In this case, the application should save the data received (from the BUFF area) and the length of the data received (from the RECLEN area) and issue another NETGET to receive the rest of the message. The application should *not* wait on the attention event between NETGETs for a single message.

Function Management Headers

The host can include function management headers (FMH) in the message. The Series/1 application becomes aware of their presence by the return code value. If you issue more than one NETGET to receive the message, only the first NETGET return code indicates the FMH. The only restriction to the formatting of the message is that the FMH(s) must be at the beginning of the message. The host's application program and the Series/1 application program decide the conventions used in formatting the message.

NETGET Coding Illustration

This coding illustration issues a NETGET instruction to receive a message or partial message and store it at address INBUFF. In addition:

- The LU is number 1 at location NETLU.
- The length of the input area is at address INBLEN.
- The length of the message or partial message received is to be stored at address COUNT.

```
      NETGET   LU=NETLU,                                          C
               BUFF=INBUFF,                                       C
               BYTES=INBLEN,                                      C
               RECLEN=COUNT
            •
            •
            •
NETLU    DATA  F'1'
INBUFF   DATA  XL80
INBLEN   DATA  F'80'
COUNT    DATA  F'0'
```

NETGET Return Codes

The positive return codes from NETGET have bit-significant values to allow for efficient analysis in the Series/1 SNA application. The bit positions have the following meaning:

```
.... .... .... ...1   Function Management Header received.
.... .... .... ..1.   End of message received.
.... .... .... .1..   Right-to-send received.
.... .... .... 1...   Response to message requested.
.... .... ...1 ....   End of transaction received.
.... .... ..1. ....   Start of transaction received.
```

The valid combinations of the bit positions are listed in the following decimal return codes:

Return Code	Condition
59	Start and end of transaction, end of message and FMH received, response requested.
58	Start and end of transaction, and end of message received, response requested.
51	Start and end of transaction, end of message and FMH received.
50	Start and end of transaction, and end of message received.
47	Start of transaction, end of message, FMH, and right-to-send received, response requested.
46	Start of transaction, end of message, and right-to-send received, response requested.
43	Start of transaction, end of message, and FMH received, response requested.
42	Start of transaction, end of message, and response requested.
39	Start of transaction, end of message, FMH, and right-to-send received.
38	Start of transaction, end of message, and right-to-send received.
35	Start of transaction, end of message, and FMH received.
34	Start of transaction, and end of message received.
33	Start of transaction, and FMH received.
32	Start of transaction received.
27	End of transaction, end of message, and FMH received, response requested.
26	End of transaction, and end of message received, response requested.
19	End of transaction, end of message, and FMH received.
18	End of transaction, and end of message received.
15	End of message, FMH, and right-to-send received, response requested.
14	End of message, and right-to-send received, response requested.

11	End of message, and FMH received, response requested.
10	End of message received, response requested.
7	End of message, FMH, and right-to-send received.
6	End of message, and right-to-send received.
3	End of message, and FMH received.
2	End of message received.
1	FMH received.
-1	Operation successful.
-7	SNA is in the process of loading or unloading and is temporarily not usable.
-8	SNA deactivating.
-9	LU is busy with another operation.
-10	Session does not exist.
-11	Instruction must be issued under program linked to $NETCMD for same program that issued NETINIT.
-12	Invalid LU number.
-13	Invalid request.
-14	SNA system error.
-15	NETTERM in progress.
-16	Session abnormally terminated by host.
-17	Status available.
-19	$SNA was never loaded or $SNA was unloaded.
-20	UNBIND HOLD received.
-21	More than two tasks already running under this LU.
-22	Session reset; CLEAR and SDT commands received.
-23	Invalid PU number (must be from 0 to 4).
-24	Selected PU not active.
-25	No messages available.
-26	Host initiated transaction.

Chapter 8. Controlling Message Exchange - NETCTL

The NETCTL instruction sends status or error information to the host application program or receives status or error information. Before you can use NETCTL, you must establish a non-passthru session with the host.

Use NETCTL:

- To accept or reject messages received from the host application program.
- To cancel a message that was partially sent to the host application program.
- To suspend or resume message transfers.
- When the Series/1 application requests the right-to-send.
- When the Series/1 application is ready to receive the next message.
- When the Series/1 application has status data to send.
- When the Series/1 application receives a "status available" (−17) return code.

The NETCTL instruction specifies a buffer (TYPE=RECV) in which the Series/1 application program receives status information. For sessions without resynchronization support, the buffer also contains the message number of a message rejected by the host.

During SNA instruction processing, the Series/1 application program can receive a return code indicating "status available" (−17). Until the Series/1 application issues NETCTL TYPE=RECV, it can issue *no other* Series/1 SNA instruction (except NETTERM).

NETCTL is invalid for passthru sessions.

You can use NETCTL to receive status regardless of which session partner has the right-to-send.

```
label         NETCTL     LU=,BUFF=,EXIT=,P1=,P2=,                    C
                         TYPE=

Required:     LU
Defaults:     TYPE=RECV
Indexable:    none
```

LU (required)
The label of a 1-word field specifying the LU and PU numbers for the session. Code the high-order byte with the PU number and the low-order byte with the LU number. The PU number can be any value from 0 to 4, with 0 implying PU #1. The LU number is the number of the LU to use for the request (on the specified PU) and can be any value from 1 to 32.

BUFF

The label of a 3-word status area. If you specify other than RECV, REJECT, or LUSTAT for the TYPE parameter, SNA ignores the BUFF parameter. The usage of the area is as follows:

- If you specify TYPE=RECV, the status received from the host returns in this area. The format of the status depends on the type of status received as indicated by the NETCTL return code. You must specify the BUFF parameter if you specify TYPE=RECV. If the return code indicates message reject (112), status message (16), or request for right-to-send (80), the status area is as follows:
  - Message reject: The first two bytes of the area are the system sense code, and the next two bytes are the user sense code.

    If you do not select message resynchronization support for the session, the last two bytes are the message number of the message rejected by the host.

    If you select message resynchronization support for the session, the message rejected by the host is always the last message sent.
  - Status message: The first two bytes of the area are the status value, and the next two bytes are the status extension field.
  - Request for right-to-send: The first two bytes of the area are the signal value, and the next two bytes are the signal extension field.
- If you specify TYPE=REJECT, you may supply the sense codes indicating the reason the host message is unacceptable. You must specify the BUFF parameter, if you specify TYPE=REJECT. The first two bytes of the area are the system sense code, and the next two bytes are the user sense code. If you do not specify the sense codes, a system sense code of X'081C' (request not executable) with a user sense code of X'0000' (no-op) is sent.

  The host message rejected is always the last message received from the host.
- If you specify TYPE=LUSTAT, you must supply the status codes to be sent. You must specify the BUFF parameter, if you specify TYPE=LUSTAT. The first two bytes of the area are the status value, and the next two bytes are the status extension field.

EXIT

The label of the Series/1 application error processing routine. Control passes to this label if any *negative* return code *less than* −1 returns to your application.

Px

Parameter naming operands. See "Using the Parameter-Naming Operands" on page 1-11 for a further description. For NETCTL, Px corresponds to the following parameters:

P1 LU

P2 BUFF

TYPE

The control operation to be performed. You must code the BUFF parameter if TYPE=RECV, REJECT, or LUSTAT. Code one of the following:

ACCEPT	Send a message acceptance to the host, if necessary, for the message received.
CANCEL	Cancel a partially transmitted message.
LUSTAT	Send status information to the host. The 4-byte status code to be sent is contained in the area specified by the BUFF parameter.
QEC	Request the host to temporarily suspend transmitting messages after the current message.
RECV (default)	Receive status information. The return code indicates the type of status information received. The data associated with the status, if applicable, returns in the area specified by the BUFF parameter.
REJECT	Send a message rejection to the host for the message received. The 4-byte sense code, containing the reason for the rejection, is specified by the BUFF parameter.
RELQ	The host can resume sending messages, after being temporarily suspended. This parameter is valid only if TYPE=QEC was previously issued.
RTR	Notify the host that the Series/1 SNA application is ready to receive the next transaction from the host.
SIG	Request the host to grant the right-to-send to the Series/1 SNA application.

Using the NETCTL Instruction

This section describes the operation of the NETCTL instruction when used in your application program. Included in this discussion is:

- Summary of protocols for NETCTL
- Receiving status
- Message verification
- Message cancellation
- Requesting the right-to-send
- Requesting suspension of message flow
- Sending status to the host
- Ready to receive.

Summary of SNA Protocols for NETCTL

Figure 8-1 lists the SNA command protocols associated with NETCTL parameters and return codes.

SNA Parameters	SNA Protocols
TYPE=ACCEPT	Positive response sent to host if previously requested by the host (RQD1 or RQD2).
TYPE=REJECT	Negative response sent to host.
TYPE=LUSTAT	SNA LUSTAT command sent to the host.
TYPE=SIG	SNA SIGNAL command sent to the host.
TYPE=RTR	SNA RTR command sent to the host.
TYPE=QEC	SNA QEC command sent to the host.
TYPE=RELQ	SNA RELQ command sent to the host.
TYPE=CANCEL	SNA CANCEL command sent to the host.
TYPE=RECV	
message reject	Negative response received from the host. Can send CANCEL.
status message	LUSTAT received from the host.
right-to-send requested	SIGNAL received from the host.
quiesced state released	RELQ received from the host.
message cancelled	CANCEL received from the host.
session termination requested	SHUTD received from the host.

Figure 8-1. SNA Protocols for NETCTL

Receiving Status

The TYPE=RECV parameter receives status from the host. The NETCTL return code indicates the type of status received. The return code values that return status information in the buffer area are "status message received" (16), "request-to-send received" (80), and "message reject received" (112).

The following lists the requests the host can make on a NETCTL TYPE=RECV operation.

Request for termination (SHUTD)

The Series/1 application can respond by sending remaining messages then terminate the session.

Request resumption of message flow (RELQ)

The Series/1 can resume sending messages (issue NETPUTs). You may receive this return code after a NETPUT or NETCTL operation results in a "session quiesced" (suspended) return code.

Request the right-to-send (SIGNAL)

The Series/1 can honor the host request for the right-to-send by issuing a NETGET instruction. Granting the right-to-send does not need to immediately follow the host's request. The Series/1 has the option of delaying the granting of the right-to-send or not granting the right at all. Refusing to grant the right-to-send may be self-defeating for the Series/1 application program, as the host application may decide to terminate the session.

Message cancelled (CANCEL)

The host cancelled the message it was sending to the Series/1 application program. The Series/1 application program on receipt of the "message cancelled" return code should discard all parts of the current message it received.

Message rejected (−RSP)

The host rejected a message that you sent. The sense data indicating the reason for the rejection returns in the buffer. Interpretation of the sense data is application dependent.

The buffer also contains, when you do not select resynchronization support, the message number of the message rejected by the host. The message number is contained in the two low-order byte positions. If you select resynchronization support, the message number does not return, because the rejected message is always the most recent message.

You can receive the above status values in combination with the following:

- The right-to-send granted to the SNA application program
- Transaction ended by host application program.

Message Verification

After receiving a message, the Series/1 application can inform the host application program of its acceptance or its rejection. If the Bind Primary Chain Response Protocol Indicator is specified as either Definite or Exception response allowed, then do the following:

- For message acceptance, code TYPE=ACCEPT.
- For message rejection, code TYPE=REJECT.

In message rejection, you can use the status buffer to convey information to the host application program concerning the reason the message was rejected.

If omitted, a system sense code of "invalid request" (−13) returns to the host application program. In the normal stream of message traffic, the Series/1 application should examine all messages as they arrive from the host. It should reject those messages that are not acceptable immediately, regardless of whether or not the host asked for message verification. Messages that are acceptable and correct normally need no specific message verification, as the successful processing of a following message (or the successful processing of a following Series/1 SNA instruction operation other than NETCTL TYPE=REJECT) implies the Series/1 application program's acceptance of the previous message.

Message Cancellation

You can cancel only messages that require an additional NETPUT or multiple NETPUTs to complete. To cancel the incomplete message, code TYPE=CANCEL. The host application discards the part of the message already received.

Note: Messages sent by a single NETPUT operation (LAST=YES) or messages that have been completed (last NETPUT had LAST=YES) cannot be cancelled.

Requesting the Right-to-Send

You can send messages to the host application program only when you have the right-to-send. If the host application has the right-to-send, you can request it by issuing NETCTL TYPE=SIG.

It is the host application's option to honor this request; it may do so immediately, delay granting the right-to-send for a time, or it may not grant it at all.

Until the host grants the right-to-send, you must continue to receive the host's messages. Notification of the transfer of the right-to-send is indicated by a return code. With the right-to-send, you can start NETPUT (sending) operations.

Requesting Suspension of Message Flow

You can request that the host application temporarily stop sending messages. You can do this by issuing NETCTL TYPE=QEC. To allow the host to resume sending messages code NETCTL TYPE=RELQ.

Sending Status Data to the Host

When you have status data to send to the host application, use NETCTL TYPE=LUSTAT. Your application places the system and user sense information to be sent in the status buffer defined by the BUFF parameter. In order to use TYPE=LUSTAT and send status information to the host, your application must have the right-to-send.

Ready to Receive

Use NETCTL TYPE=RTR to inform the host application you are ready to receive a transaction.

Use TYPE=RTR when no transaction is in progress on the session.

Note: If the Series/1 application rejected a previous host transaction, the application must issue either NETCTL TYPE=RTR or NETGET while no transactions are active and before the host can begin another transaction. The host can still send messages while a transaction is active and the host has the right-to-send.

NETCTL Illustrations

The illustrations presented here show variations in the use of the NETCTL instruction to control message exchange.

Receiving Status from Host

This coding illustration shows a NETCTL issued to receive the status condition and store the status data (if any) at address STATUS.

```
      NETCTL   LU=NETLU,                                          C
               TYPE=RECV,                                         C
               BUFF=STATUS
            •
            •
            •
NETLU    DATA   F'1'
STATUS   DATA   3F'0'
```

Rejecting a Message

This coding illustration shows a NETCTL issued from PU #2, LU #1 to reject the message received.

```
         NETCTL   LU=NETLU,                                     C
                  TYPE=REJECT
            •
            •
            •
NETLU    DATA   X'0201'
```

Sending Status to Host

This coding illustration shows a NETCTL issued to send status to the host with the status data to be sent from address STATUS.

```
         NETCTL   LU=NETLU,                                     C
                  TYPE=LUSTAT,                                  C
                  BUFF=STATUS
            •
            •
            •
NETLU    DATA   F'1'
STATUS   DATA   3F'0'
```

NETCTL Return Codes

The positive return codes from NETCTL TYPE = RECV have bit-significant values to allow for efficient analysis in the Series/1 SNA application. The bit positions have the following meaning:

....1 End of transaction received
....1. Right-to-send received

The following values return in combination with the above bit-significant information:

16	Status message received.
32	Message being received from host canceled.
48	Session termination request received.
80	Request for right-to-send received.
96	Host permission to resume sending received.
112	Message sent to host rejected.

The valid combinations of the values and bit positions are listed in the following decimal return codes.

Return Code	**Condition**
112	Negative response received.
96	RELQ received.
80	SIGNAL received.
48	SHUTDOWN received.
34	CANCEL with CD received.
33	CANCEL with EB received.
32	CANCEL received.
18	LUSTAT with CD received.
17	LUSTAT with EB received.
16	LUSTAT received.
2	CHANGE DIRECTION received.
1	END BRACKET received.
-1	Operation successful.
-7	SNA is in the process of loading or unloading and is temporarily not usable.
-8	SNA deactivating.
-9	LU is busy with another operation.
-10	Session does not exist.

-11 Instruction must be issued under program linked to $NETCMD for same program that issued NETINIT.

-12 Invalid LU number.

-13 Invalid request.

-14 SNA system error.

-15 NETTERM in progress.

-16 Session abnormally terminated by host.

-17 Status available.

-18 Session quiesced.

-19 $SNA was never loaded or $SNA was unloaded.

-20 UNBIND HOLD received.

-21 More than two tasks already running under this LU.

-22 Session reset; CLEAR and SDT commands received.

-23 Invalid PU number.

-24 Selected PU not active.

-25 Not right-to-send.

-26 No status available.

In addition to the previously listed NETCTL return codes, Figure 8-2 lists these return codes by the type of NETCTL TYPE= parameter specified.

Return Code	Condition	NETCTL Operation
112 96 80 48 34 33 32 18 17 16 2 1	Negative response received RELQ received SIGNAL received SHUTDOWN received CANCEL with CD received CANCEL with EB received CANCEL received LUSTAT with CD received LUSTAT with EB received LUSTAT received CHANGE DIRECTION received END BRACKET received	Only NETCTL TYPE=RECV can generate these return codes
-1 -7 -8 -9 -10 -11 -12 -13 -14 -15 -16 -19 -21	Operation successful SNA is in the process of loading or unloading and is temporarily not usable SNA deactivating LU is busy with another operation Session does not exist Instruction must be issued under program linked to $NETCMD for same program that issued NETINIT Invalid LU number Invalid request SNA system error NETTERM in progress Session abnormally terminated by host $SNA was never loaded or $SNA was unloaded More than two tasks already running under this LU	All NETCTL TYPEs can generate these return codes
-17 -20 -22	Status available UNBIND HOLD received Session reset; CLEAR and SDT commands received	All NETCTL TYPEs except TYPE=RECV can generate this return code

Figure 8-2 (Part 1 of 2). NETCTL TYPE= Return Codes

Return Code	Condition	NETCTL Operation
-18 -23 -24 -25	Session quiesced Invalid PU number Selected PU not active Not right-to-send	Only NETCTL TYPE=CANCEL, RTR, and LUSTAT can generate these return codes
-26	No status available	Only NETCTL TYPE=RECV can generate this return code

Figure 8-2 (Part 2 of 2). NETCTL TYPE= Return Codes

Chapter 9. Terminating a Session - NETTERM

The NETTERM instruction releases the logical communication path previously established between session partners by the NETINIT instruction. NETTERM terminates the session and releases the Series/1 resources used for the session.

System resources released by NETTERM become available for the establishment of another session.

Use NETTERM when:

- You no longer desire dialogue with the session partner.
- The host requested termination.
- A severe error condition causes abnormal session termination.
- You want to terminate a passthru connection between two remote LUs.

Return codes inform you of the host request for termination and of abnormal session termination.

Either the host or the Series/1 application can request session termination at any time. Series/1 SNA support handles the protocol associated with NETTERM.

No messages can be exchanged on a session in which one of the session partners issued a successful NETTERM. Series/1 SNA support rejects messages received from the host after issuance of NETTERM.

The following is the coding syntax and parameter descriptions of the NETTERM instruction.

```
label           NETTERM       LU=,EXIT=,HOLD=,TYPE=,P1=

Required:       LU
Defaults:       HOLD=NO,TYPE=NORMAL
Indexable:      none
```

LU (required)
The label of a 1-word field specifying the LU and PU numbers for the session. Code the high-order byte with the PU number and the low-order byte with the LU number. The PU number can be any value from 0 to 4, with 0 implying PU #1. The LU number is the number of the LU to use for the request (on the specified PU) and can be any value from 1 to 32.

EXIT
The label of the Series/1 application error processing routine. Control passes to this label if any *negative* return code *less than* −1 returns to your application.

HOLD

Whether to release or keep session resources if the host issues a subsequent BIND command. Use this parameter only upon receipt of an UNBIND HOLD from the host. Specify one of the following:

NO (default) Terminate the session and release all resources.

YES Process the UNBIND-HOLD command and retain the session resources required to handle the subsequent BIND command sent from the host.

TYPE

Specifies the type of session termination requested by the Series/1 application. Specify one of the following:

NORMAL (default)
The Series/1 application requests the host LU to terminate the session between itself and the host LU.

QUICK The Series/1 application requests VTAM to terminate the session between itself and the host LU.

IMMED The Series/1 application requests that the Series/1 SNA support terminate the session between itself and the host LU. The Series/1 SNA support does not wait for any response from the host.

Note: Use of the TYPE=IMMED parameter might not be supported by all host systems. For more details, see "Unconditional Session Termination" on page 9-3

For passthru sessions, only TYPE=NORMAL is supported.

Px

Parameter naming operand. See "Using the Parameter-Naming Operands" on page 1-11 for further description. For NETTERM, Px corresponds to these parameters: P1 and LU.

Using the NETTERM Instruction

This section describes the NETTERM instruction when used in your application program.

You can initiate session termination as an *orderly termination* or as an *unconditional termination*.

Orderly Session Termination

NETTERM or the host can call for the orderly termination of a session. You can only call for orderly termination from your application program. When session termination is initiated by the host, the Series/1 SNA support passes a return code to the Series/1 application program, indicating "status available" (−17). A subsequent NETCTL TYPE=RECV return code is "shutdown received" (48). The application should then send any remaining messages to the host and then issue NETTERM, TYPE=NORMAL.

The Series/1 SNA application can also call for an orderly termination by issuing a NETTERM, TYPE=NORMAL operation before the host sends SHUTD. In this case, Series/1 SNA support sends the SNA Request Shutdown (RSHUTD) command to prompt the host to terminate the session. (Note the RSHUTD does not request the host to send SHUTD. The host sends UNBIND.)

Unconditional Session Termination

Both the Series/1 application and the host can call for unconditional termination, which indicates a severe error condition (such as lack of resources) on the session.

A "session abnormally terminated" (−16) return code indicates that the host has called for unconditional session termination. The Series/1 application program cannot send any messages to the host when unconditional termination is in effect. Following the receipt of the unconditional termination return code, the application should issue NETTERM to free resources held to support the session.

The Series/1 application program can call for unconditional session termination by issuing a NETTERM with either TYPE=QUICK or TYPE=IMMED. If you specify TYPE=QUICK, the Series/1 SNA support sends a TERM-SELF SNA command to the host. This command goes to the SSCP component of VTAM; VTAM then terminates the session with the host LU by sending an UNBIND to the Series/1.

If you specify TYPE=IMMED, the Series/1 SNA support sends an UNBIND command to the host and frees the Series/1 resources immediately. No further messages or SNA commands are accepted from the host. Some host systems may not process the receiving of an UNBIND from a secondary LU. For example, MVS VTAM must be at ACF Release 1.3 or later for this option to operate correctly. Consult your host system programmer to determine if TYPE=IMMED can be used.

UNBIND HOLD Support

The primary LU issues an UNBIND HOLD command to terminate its session with a secondary without releasing the secondary LU's resources. Another primary LU can then issue a BIND command to the secondary LU and establish a new session.

When the Series/1 receives an UNBIND HOLD command from the primary LU, the following occurs:

- The attention event, if specified, is posted. This action requires the Series/1 application to issue a NETGET instruction. The NETGET instruction completes with a return code of −20 (UNBIND HOLD was received).
- If a NETINIT is in progress at the Series/1, the NETINIT completes with a good return code. The Series/1 application receives notification on its next operation that the host received the UNBIND HOLD.
- If a NETCTL, NETGET, or NETPUT instruction is in progress, the operation completes with a −20 return code (UNBIND HOLD was received).
- The area defined by NETINIT's ERRCODE parameter is updated as follows:
  - Byte 0 contains the SNA instruction type
  - Byte 1 contains an "error function" of 12 (indicating session termination)
  - Bytes 2−3 contain a return code (−20) indicating the UNBIND HOLD was received.

After executing NETTERM HOLD=YES, you can issue, *for that session only*:

- The NETTERM instruction with HOLD=NO to release all the resources retained for that session
- The NETINIT instruction with the HOLDLU parameter in anticipation of the second BIND.

A NETINIT instruction specifying the HOLDLU parameter indicates that the NETINIT is for a session for which there was a previously executed NETTERM instruction specifying HOLD=YES. Not specifying HOLDLU indicates that the NETINIT instruction is for a new session.

Note: Series/1 SNA does not support message resynchronization for sessions using UNBIND HOLD.

NETTERM - Coding Illustration

The following coding illustration shows session termination with the NETTERM instruction. The LU address for the terminated session is at address NETLU.

```
         NETTERM LU=NETLU,                                          C
           •
           •
           •
NETLU    DATA   F'1'
```

NETTERM Return Codes

The positive return codes from NETTERM have bit-significant values to allow for efficient analysis in the Series/1 SNA application. The bit positions have the following meaning:

```
.... .... .... ...1     Message from host rejected during termination.
.... .... .... ..1.     Message to host rejected during termination.
.... .... .... .1..     Message to host aborted during termination.
.... .... .... 1...     Message from host aborted during termination.
```

The valid combinations of the bit positions are listed in the following decimal return codes:

Return Code	Condition
9	CANCEL received during NETTERM and -RSP sent during NETTERM.
8	CANCEL received during NETTERM.
7	CANCEL sent during NETTERM and -RSP received during NETTERM and -RSP sent during NETTERM.
6	CANCEL sent during NETTERM and -RSP received during NETTERM.
5	CANCEL sent during NETTERM and -RSP sent.
4	CANCEL sent during NETTERM.
3	-RSP received during NETTERM and -RSP sent during NETTERM.
2	-RSP received during NETTERM.
1	-RSP sent during NETTERM.
-1	Operation successful.
-7	SNA is in the process of loading or unloading and is temporarily not usable.
-10	Session does not exist.
-11	Instruction must be issued under program linked to $NETCMD for the same program that issued NETINIT.
-12	Invalid LU number.
-14	SNA system error.
-15	NETTERM in progress.
-16	Session abnormally terminated.
-19	$SNA was never loaded or $SNA was unloaded.
-20	UNBIND HOLD received.
-23	Invalid PU number.
-24	Selected PU not active.
-25	No UNBIND HOLD received.

Chapter 10. Message Recovery and Resynchronization

Message recovery and resynchronization ensures the integrity of message flow during a session and across the restart of a session. The Series/1 SNA application can select message resynchronization support for a session when specifying the NETINIT RESYNC=YES or INIT parameters. If the application specifies message resynchronization, Series/1 SNA support keeps track of messages transmitted and received during the session by saving sequence numbers of the messages.

For example, if the communications line goes down during a transmission, resynchronization enables the Series/1 SNA support to notify the host that data may have been lost. Series/1 SNA support notifies the host by responding to the host Set and Test Sequence Numbers (STSN) command the next time the session is established.

Sequence Numbers

During a session, the messages transmitted between the host subsystem and the Series/1 SNA application are numbered. The sender assigns numbers consecutively, beginning with 1. The host assigns numbers to messages sent to the Series/1 application. Series/1 SNA support assigns numbers to messages sent to the host.

When you establish the session with resynchronization support, the sequence number of the last message flowing in each direction is saved for resynchronization. The Series/1 SNA application does not need to be concerned with sequence numbers on resynchronization support sessions. The sequence numbers are internal to the Series/1 SNA support.

For sessions without resynchronization support, the sequence numbers optionally return to the Series/1 SNA application on send (NETPUT) operations. You can use these numbers to correlate negative responses from the host to transmitted messages.

Message Resynchronization Data Set

For sessions established with resynchronization support (NETINIT RESYNC=YES or INIT), Series/1 SNA maintains the resynchronization data necessary for recovery on a Series/1 disk or diskette data set or in Series/1 main storage. SNA support does not use the resynchronization data unless you establish the session with resynchronization support.

When you specify disk/diskette storage (NETINIT RTYPE=DISK), Series/1 SNA support saves the message resynchronization data in the data set specified on the session establishment operation (NETINIT RDSCB=). You must previously allocate the volume and data set to contain the resynchronization data with a length of 256 bytes. The Series/1 SNA application has the responsibility of opening and closing the data set. Series/1 SNA support saves the resynchronization data for the session at the relative record corresponding to the Series/1 LU number for the session.

When you specify main storage (NETINIT RTYPE=STG), Series/1 SNA support maintains the resynchronization data in an area specified on the session establishment operation. The data saved in main storage is identical in content and format to that stored on disk/diskette.

Resynchronization Data Set Contents

Figure 10-1 describes the contents of the resynchronization data set (disk and Series/1 storage) when Series/1 SNA support starts the resynchronization process.

Field	Contents	When updated
Transmit Sequence Number	The sequence number of the last BIU (End Chain) of the last message transmitted to the host.	Saved after the last (EC) buffer in a chain goes to the base SNA support. Restored to the old sequence number (prior chain's ending sequence number) if the application receives a negative response from the host.
Receive Sequence Number	The sequence number of the last BIU (End Chain) of the last message received from the host *and* acknowledged by the Series/1 SNA application with a positive response or a reply.	Saved at the time that the Series/1 SNA determines that a positive response or reply should be sent to the host. Unchanged if a negative response is to be sent to the host.
Transmit Indicator (TI)	A flag indicating that the Series/1 SNA is processing a transmit operation.	Set when the first (or only) NETPUT operation in a chain is detected. Reset when the last (EC) buffer in a chain goes to the base SNA support.
Receive Indicator (RI)	A flag indicating that the Series/1 SNA is processing a receive operation.	Set when the first (or only) NETGET operation in a chain is detected. Reset when a reply or positive response is to be transmitted to the host.

Figure 10-1 (Part 1 of 2). Resynchronization Data Set Contents

Field	Contents	When updated
Pending Reply Indicator (PRI)	A flag indicating that the Series/1 SNA has sent a complete chain to the host and is waiting for an acknowledgment (reply or response) from the host.	Set when the last (EC) buffer goes to the base SNA support to be transmitted to the host. Reset when a reply or response is received from the host.

Figure 10-1 (Part 2 of 2). Resynchronization Data Set Contents

Considerations for Disk and Main Storage

The following considerations apply to disk/diskette and main storage retention of the resynchronization data:

- Retention on disk/diskette is the more reliable method. Retention on disk/diskette assures that the resynchronization data will be maintained across a power failure or system failure that might destroy Series/1 main storage contents. However, disk/diskette access and data transfer time adds significant overhead to each transaction.
- Retention in Series/1 main storage minimizes the overhead to each transaction. Retention in Series/1 main storage is subject to loss due to power failures or system failures that might destroy main storage contents. If this happens, message resynchronization is not possible. This could result in the loss of one message sent to the host and the duplication or loss of one message sent by the host.

Recovery/Resynchronization Protocols

The host sends the SNA Set and Test Sequence Numbers (STSN) command to resynchronize the sequence numbers for the session in the course of recovery procedures. You can use the command to resynchronize either the host-to-Series/1 (PLU-to-SLU) flow, the Series/1-to-host (SLU-to-PLU) flow, or both flows. The command describes the resynchronization action for the flows and may carry the host sequence numbers for the flows.

The host may set the resynchronization action as described below independently for each flow.

Set The host must set the sequence number for the flow to the value provided on the STSN command.

Sense The response to the STSN command must contain the sequence number maintained for the flow.

Set-and-Test The host must set the sequence number for the flow to the value provided on the STSN command, and the response to the STSN command must indicate the results of the comparison of the sequence number in the command and the sequence maintained for the flow.

Ignore Take no action on the sequence number for the flow.

When the host subsystem starts resynchronization procedures, the Series/1 SNA support receives and responds to the STSN command sent by the host using the sequence numbers saved for the session in the resynchronization data. A return code from the NETINIT instruction informs the Series/1 SNA application of the session restart state.

If the return code indicates message recovery is necessary, the SNA application should perform the recovery procedures previously established with the host subsystem program. (See Appendix D, "Host Subsystem Considerations" for more information about subsystem recovery procedures.)

In general, the following considerations apply for sessions established with resynchronization support:

- The resynchronization data compares the sequence numbers.
- If the sequence numbers agree, the session is successfully resynchronized.
- If the "send" (SLU-to-PLU) sequence numbers do not match, this usually indicates that the host program did not receive a message sent by Series/1 SNA in the previous session.
- If the "receive" (PLU-to-SLU) sequence numbers do not match, this usually indicates that Series/1 SNA did not receive a message sent by the host in the previous session.
- The Series/1 SNA application is notified of the session restart state.

The following considerations generally apply for sessions established without resynchronization support:

- The resynchronization data is not used.
- The host may still attempt to reset the sequence numbers to be used on the session.
- The Series/1 SNA application is not notified of the session restart state.

To start a session when the previous state of the session is unknown, you can select the NETINIT RESYNC=INIT option. In this mode, the Series/1 SNA support accepts any restart state from the host. However, the loss of any messages on the previous session are not reported or recovered.

Series/1 Processing of the STSN Command

Figure 10-2 on page 10-6, Figure 10-3 on page 10-13, and Figure 10-4 on page 10-20 summarize the Series/1 SNA processing of the STSN command.

The columns in these three figures are:

Reference Number (RN)
A number corresponding to the explanation of the entry in the figures.

STSN Request
The action selected by the host to be taken when the STSN command is processed. The STSN "action code" bit setting is shown. For certain action codes, the host sends its sequence number maintained for the flow from the previous session.

Sequence Numbers

The results of the comparison of the sequence number received on the STSN to the sequence number in the resynchronization data.

The following terms describe the results of the sequence number comparison:

- INIT — The Series/1 SNA application specifies that the sequence numbers in the resynchronization data are to be initialized by Series/1 SNA based on the NETINIT instruction (RESYNC=INIT).
- COLD START — Either the host did not send a STSN, or the host sequence numbers on both flows on the STSN was X'0000'.
- EQUAL — The host sequence number for the flow is the same as the sequence number for the flow in the resynchronization data.
- UNEQUAL — The host sequence number for the flow is not the same as the sequence number for the flow in the resynchronization data.
- N/A — The sequence number relationship does not affect the processing.

Resynchronization Data Indicators

The settings of the indicators in the resynchronization data at the time that the STSN is received from the host. The following abbreviations are used:

- TI — Transmit Indicator
- RI — Receive Indicator
- PRI — Pending Response Indicator

Series/1 SNA Action

The applicable processing by the Series/1 SNA support.

Return Code

The return code for the particular message flow. The return code actually passed to the application is a composite return code, indicating the results of message synchronization for both of the message flows. (The actual return code passed to the application is determined by the composite return code table shown in Figure 10-5 on page 10-22).

STSN Response

The response to STSN made by the Series/1 SNA support. The STSN "result code" bit setting is shown. For certain result codes, Series/1 SNA sends its sequence number maintained for the flow from the previous session.

STSN Processing for Sessions with Resynchronization Support

Figure 10-2 and Figure 10-3 on page 10-13 describe processing for SLU-to-PLU and PLU-to-SLU message flows during sessions established with resynchronization support. (Sessions without resynchronization support are described in "STSN Processing for Sessions without Resynchronization Support" on page 10-20.) Following each figure is an explanation of the message flow for various cases.

SLU-to-PLU Message Flow

Figure 10-2 shows the SLU-to-PLU message flow during the session. Following this figure is an explanation of the message flow.

RN	STSN Request	Sequence Numbers	Indicators	Series/1 SNA Action	Return Code	STSN Response
1	Ignore (B'00')	N/A	N/A	none	SYNCHED	ignore (B'01')
2	set (B'01') or set&test (B'11')	INIT	N/A	Set transmit sequence number to host value. Reset transmit indicator and pending reply indicator	SYNCHED	Ignore (B'01') or match (B'01') host sequence number returned
3	set (B'01') or set&test (B'11')	EQUAL	TI = reset PRI = reset	No change to resynch data	SYNCHED	Ignore (B'01') or match (B'01') transmit sequence number returned
4	set (B'01') or set&test (B'11')	EQUAL	TI = reset PRI = set	Reset pending response indicator.	SYNCHED	Ignore (B'01') or match (B'01') transmit sequence number returned
5	set (B'01') or set&test (B'11')	EQUAL	TI = set PRI = reset	Reset transmit indicator.	MESSAGE TO HOST LOST	Ignore (B'01') or match (B'01') transmit sequence number returned

Figure 10-2 (Part 1 of 3). STSN Processing for the SLU-to-PLU Message Flow

RN	STSN Request	Sequence Numbers	Indicators	Series/1 SNA Action	Return Code	STSN Response
6	set (B'01') or set&test (B'11')	UNEQUAL	TI = reset PRI = reset	No change to resynch data	STSN ERROR	Ignore (B'01') or invalid (B'10')
7	set (B'01') or set&test (B'11')	UNEQUAL	TI = reset PRI = set	Set transmit sequence number to host value. Reset pending response indicator	MESSAGE TO HOST LOST	Ignore (B'01') or mismatch (B'11') transmit sequence number returned
8	set (B'01') or set&test (B'11')	UNEQUAL	TI = set PRI = reset	No change to resynch data	STSN ERROR	Ignore (B'01') or invalid (B'10')
9	set (B'01') or set&test (B'01')	COLD START	TI = reset PRI = reset	Set transmit sequence number to X'0000'	HOST COLD START	Ignore (B'01') or match (B'01') or mismatch (B'11') transmit sequence number returned
10	set (B'01') or set&test (B'01')	COLD START	TI = reset PRI = set	Set transmit sequence number to X'0000' Reset pending response indicator.	POSSIBLE MESSAGE TO HOST LOST + HOST COLD START	Ignore (B'01') or match (B'01') or mismatch (B'11') transmit sequence number returned

Figure 10-2 (Part 2 of 3). STSN Processing for the SLU-to-PLU Message Flow

RN	STSN Request	Sequence Numbers	Indicators	Series/1 SNA Action	Return Code	STSN Response
11	set (B'01') or set&test (B'01')	COLD START	TI = set PRI = reset	Set transmit sequence number to X'0000' Reset transmit indicator	MESSAGE TO HOST LOST + HOST COLD START	Ignore (B'01') or match (B'01') or mismatch (B'11') transmit sequence number returned
12	sense (B'10')	N/A	N/A	No change to resynch data	SYNCHED	OK (B'11') and sequence number returned

Figure 10-2 (Part 3 of 3). STSN Processing for the SLU-to-PLU Message Flow

SLU-to-PLU Flow Explanation

The reference numbers in Figure 10-2 on page 10-6 correspond to the numbered descriptions below.

1 The host selected the "ignore" option on the STSN for this message flow. Series/1 SNA takes no action on this STSN except to create a SYNCHRONIZED return code for this flow and an STSN result of "ignore" (B'01').

2 The host selected the "set" or "set&test" option on the STSN. The Series/1 SNA application selected to have the Series/1 SNA support initialize the contents of the resynchronization data during initialization. Series/1 SNA sets the transmit sequence number to the value contained in the STSN for this message flow. Series/1 SNA resets the transmit indicator and the pending reply indicator. Series/1 SNA generates a SYNCHRONIZED return code for this message flow.

The STSN response depends upon the STSN option selected by the host. The host always sends a result code of "ignore" (B'01') for the "set" option and handles the response to "set&test" as follows:

a Sets the result code to "match"

b Returns the host sequence number.

3 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is the same as the transmit sequence number in the resynchronization data. Both the transmit and pending reply indicators are off.

This is the normal case, where both the Series/1 SNA application and the host are synchronized and no replies or responses are outstanding. SNA generates a SYNCHRONIZED return code for this message flow.

The STSN response depends upon the STSN option selected by the host. The host always sends a result code of "ignore" (B'01') for the "set" option and handles the response to "set&test" as follows:

a The STSN result code is "match."

b The transmit sequence number from the resynchronization data returns in the STSN response.

4 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is the same as the transmit sequence number in the resynchronization data. However, the pending reply indicator is set in the resynchronization data.

The processing of the previous session, which left this condition, appears below. The sequence numbers shown are examples of those maintained by Series/1 SNA and by the host subsystem.

Previous Processing	Application Sequence Number	TI	PRI	Host Sequence Number
1. Both ends synchronized	10	off	off	10
2. Application issues NETPUT	10	on	off	10
3. Series/1 SNA sends last buffer in the chain for the NETPUT	11	off	on	10
4. Host processes the message and sends reply	11	off	on	11
5. Link breaks before Series/1 SNA sees reply.	11	off	on	11

In this case, the host processed the entire message sent by the application, but the application does not know it because the reply was lost. The SYNCHRONIZED return code for the SLU-to-PLU flow on this NETINIT informs the application that the last NETPUT message was processed. (Contrast this case with number 7, below, where the host did not process the message.)

Series/1 SNA resets the PRI indicator.

The STSN response depends upon the STSN option selected by the host. The host always sends a result code of "ignore" (B'01') for the "set" option and handles the response for "set&test" as follows:

a The STSN result code is "match."

b The transmit sequence number from the resynchronization data returns in the STSN response.

5 The host selected the "set" or the "set&test" option on the STSN. The sequence number on the STSN is the same as the transmit sequence number in the resynchronization data. However, the host sets the transmit indicator in the resynchronization data.

The processing of the previous session, which left this condition, appears below. The sequence numbers shown are examples of those maintained by Series/1 SNA and by the host subsystem.

Previous Processing	Application Sequence Number	TI	PRI	Host Sequence Number
1. Both ends synchronized	10	off	off	10
2. Application issues NETPUT	10	on	off	10
3. Link breaks before Series/1 SNA sends end chain.				

In this case, the host has not processed a complete message sent by the NETPUT instruction. The application does not know that the message did not make it to the host. A return code indicating that all of the message was not sent to the host (MESSAGE TO HOST LOST) is generated for this flow.

Series/1 SNA resets the TI indicator.

The STSN response depends upon the STSN option selected by the host. The host always sends a result code of "ignore" (B'01') for the "set" option and handles the response for "set&test" as follows:

a The STSN result code is "match."

b The transmit sequence number from the resynchronization data returns in the STSN response.

6 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is not the same as the transmit sequence number in the resynchronization data. Both the transmit and pending reply indicators are off. This case is treated as an error, since no messages were in process, yet the host and the application are out of synchronization.

The resynchronization data is not updated. The return code for this flow is STSN ERROR. The STSN result code for "set" on this flow is "ignore" since that is the only valid result code. The STSN result code for "set&test" is "invalid" to signal the host of the message synchronization problem.

7 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is not the same as the transmit sequence number in the resynchronization data. The pending reply indicator is on.

The processing of the previous session, which left this condition, appears below. The sequence numbers shown are examples of those maintained by Series/1 SNA and by the host subsystem.

Previous Processing	Application Sequence Number	TI	PRI	Host Sequence Number
1. Both ends synchronized	10	off	off	10
2. Application issues NETPUT	10	on	off	10
3. Series/1 SNA sends last buffer in the chain for the NETPUT	11	off	on	10
4. Link breaks before host processes the message				

In this case, Series/1 SNA sent the entire chain to the host, but an error occurred and the link broke before the host had a chance to process it. Note that the host sequence number was not updated.

The "message to host lost" return code for this flow informs the application that the last NETPUT failed. The resynchronization data is updated to the host number, and the pending reply indicator is reset.

The STSN response depends upon the STSN option selected by the host. The host always sends a result code of "ignore" (B'01') for the "set" option and handles the response for "set&test" as follows:

a The STSN result code is "mismatch."

b The transmit sequence number from the resynchronization data returns in the STSN response.

8 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is not the same as the transmit sequence number in the resynchronization data. The transmit indicator is set in the resynchronization data.

This is the same as 6 on page 10-10. The numbers should match, but the host and our application are out of synchronization.

The resynchronization data is not updated. The return code for this flow is STSN ERROR. The STSN result code for "set" on this flow is "ignore." The STSN result code for "set&test" is "invalid" to signal the host of the message synchronization problem.

9 The host selected the "set" or "set&test" option on the STSN, with the sequence number of each flow being X'0000', indicating a host cold start. Neither the transmit indicator nor the pending reply indicator is set, meaning that no message transmission to the host was in process.

The transmit sequence number is set to X'0000' to match the host. A SYNCHRONIZED return code is generated for this flow, indicating the host cold start. (This processing also occurs if no STSN is received.)

The STSN response depends upon the STSN option selected by the host. The host always sends a result code of "ignore" (B'01') for the "set" option and handles the response for "set&test" as follows:

a If the application chooses to initialize the contents of the resynchronization data, SNA uses the transmit sequence number X'0000'; otherwise, it uses the transmit sequence number from the resynchronization data.

b If the transmit sequence number is X'0000', the result code is "match"; otherwise, it is "mismatch."

c The transmit sequence number returns in the STSN response.

10 The host selected the "set" or "set&test" option on the STSN. The sequence number of each flow is X'0000', indicating a host cold start. The pending reply indicator is set, meaning that a complete chain was sent to the host, but no reply was received.

There is no way to tell whether the host processed the message before the outage that caused the cold start occurred. The return code generated for this flow indicates that the message sent to the host may have been lost and that the host was cold started. You must determine through external means whether the message was processed by the host. (This processing also occurs if no STSN is received.)

The transmit sequence number is set to X'0000', and the pending response indicator is reset.

The STSN response depends upon the STSN option selected by the host. The host always send a result code of "ignore" (B'01') for the "set" option and handles the response for "set&test" as follows:

a If the application chooses to initialize the contents of the resynchronization data, it uses the transmit sequence number X'0000'; otherwise, it uses the transmit sequence number from the resynchronization data.

b If the transmit sequence number is X'0000', the result code is "match"; otherwise, it is "mismatch."

c The transmit sequence number returns in the STSN response.

11 The host selected the "set" or "set&test" option on the STSN. The sequence number of each flow is X'0000', indicating a host cold start. The transmit indicator is set, meaning that Series/1 SNA was processing a NETPUT operation when the outage occurred but that the end chain RU was not yet transmitted by Series/1 SNA. The return code generated for this flow indicates that a message sent to the host was lost, and that the host was cold started.

The transmit sequence number is set to X'0000', and the transmit indicator is reset. (This processing also occurs if no STSN is received.)

The STSN response depends upon the STSN option selected by the host. The host always sends a result code of "ignore" (B'01') for the "set" option and handles the response for "set&test" as follows:

a If the application chooses to initialize the contents of the resynchronization data, it uses the transmit sequence number X'0000'; otherwise, it uses the transmit sequence number from the resynchronization data.

b If the transmit sequence number is X'0000', the result code is "match"; otherwise, it is "mismatch."

c The transmit sequence number returns in the STSN response.

12 The host selected the "sense" option on the STSN. Series/1 SNA responds to the STSN with an "ok" result code (B'11') and includes the sequence number in the response. The sequence number is determined as follows:

a If the application wants Series/1 SNA to initialize the resynchronization data contents, a sequence number of X'0000' returns to the host.

b Otherwise, it uses the transmit sequence number from the resynchronization data.

PLU-to-SLU Message Flow

Figure 10-3 shows the PLU-to-SLU message flow during the session. Following the figure is an explanation of this message flow.

RN	STSN Request	Sequence Numbers	Indicators	Series/1 SNA Action	Return Code	STSN Response
1	ignore (B'00')	N/A	N/A	none	SYNCHED	ignore (B'01')
2	set (B'01') or set&test (B'11')	INIT	N/A	Set receive sequence number to host value. Reset receive indicator	SYNCHED	ignore (B'01') or match (B'01') host sequence number returned

Figure 10-3 (Part 1 of 3). STSN Processing for the PLU-to-SLU Message Flow

RN	STSN Request	Sequence Numbers	Indicators	Series/1 SNA Action	Return Code	STSN Response
3	set (B'01') or set&test (B'11')	EQUAL	RI = reset	No change to resynch data	SYNCHED	ignore (B'01') or match (B'01') and receive sequence number returned
4	set (B'01') or set&test (B'11')	EQUAL	RI = set	Reset receive indicator.	SYNCHED	ignore (B'01') or match (B'01') and receive sequence number returned
5	set (B'01') or set&test (B'11')	UNEQUAL	RI = reset	Set receive sequence number to host value.	MESSAGE FROM HOST LOST	ignore (B'01') or match (B'01') and receive sequence number returned
6	set (B'01') or set&test (B'11')	UNEQUAL	RI = set	Set receive sequence number to host value. Reset receive indicator.	PARTIAL MESSAGE FROM HOST LOST	ignore (B'01') or mismatch (B'11') and receive sequence number returned

Figure 10-3 (Part 2 of 3). STSN Processing for the PLU-to-SLU Message Flow

RN	STSN Request	Sequence Numbers	Indicators	Series/1 SNA Action	Return Code	STSN Response
7	set (B'01') or set&test (B'11')	COLD START	RI = reset	Set receive sequence number to X'0000'	SYNCHED + HOST COLD START	ignore (B'01') or match (B'01') or mismatch (B'11') receive sequence number returned
8	set (B'01') or set&test (B'11')	COLD START	RI = set	Set receive sequence numbers to X'0000'. Reset receive indicator.	MESSAGE FROM HOST LOST + HOST COLD START	ignore (B'01') or match (B'01') or mismatch (B'11') receive sequence number returned
9	sense (B'10')	N/A	N/A	No change to resynch data	SYNCHED	OK (B'11') and sequence number returned

Figure 10-3 (Part 3 of 3). STSN Processing for the PLU-to-SLU Message Flow

PLU-to-SLU Flow Explanation

The reference numbers in Figure 10-3 correspond to the numbered descriptions below.

1 The host selected the "ignore" option on the STSN for this message flow. Series/1 SNA takes no action on this STSN except to create a SYNCHRONIZED return code for this flow and an STSN result code of "ignore" (B'01').

2 The host selected the "set" or "set&test" option on the STSN. The Series/1 SNA application selected to have Series/1 SNA initialize the contents of the resynchronization data during initialization. Series/1 SNA sets the receive sequence number to the value contained in the STSN for this message flow and resets the receive indicator. Series/1 SNA generates a SYNCHRONIZED return code for this message flow.

The STSN response generated for this flow depends upon the STSN option selected by the host. The host always returns a result code of "ignore" (B'01') to the "set" option and handles the response to "set&test" as follows:

a The result code is set to "match."

b The host sequence number returns in the STSN response.

3 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is the same as the receive sequence number in the resynchronization data. The receive indicator is off.

This is the normal case, where both the Series/1 SNA application and the host are synchronized and no replies or responses are outstanding. A SYNCHRONIZED return code is generated for this message flow.

The STSN response generated for this flow depends upon the STSN option selected by the host. The host always returns a result code of "ignore" (B'01') to the "set" option and handles the response for "set&test" as follows:

a The STSN result code is "match."

b The receive sequence number from the resynchronization data returns in the STSN response.

4 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is the same as the receive sequence number in the resynchronization data. However, the receive indicator is set in the resynchronization data. The processing of the previous session, which left this condition, appears in the example that follows. The sequence numbers shown are examples of those maintained by Series/1 SNA and by the host subsystem.

Previous Processing	Application Sequence Number	RI	PRI	Host Sequence Number
1. Both ends synchronized	10	off		10
2. Host sends chain of three RUs.	10	off		13
3. Application issues NETGET	10	on		13
4. Link breaks before application gets all of chain.	10	on		13
5. Message purged by host operator.	10	on		10

In this case, the application issued one or more NETGET operations but did not receive the entire chain before the link broke. A SYNCHRONIZED return code for the PLU-to-SLU flow informs the application that the last host message was purged and will not be sent again by the host.

Contrast this example with case number 6, where the chain was not purged and where the host may send the message again.

Series/1 SNA resets the RI indicator and generates a SYNCHRONIZED return code for this message flow.

The STSN response generated for this flow depends upon the STSN option selected by the host. The host always returns a result code of "ignore" (B'01') to the "set" option and handles the response to "set&test" as follows:

a The STSN result code is "match."

b The receive sequence number from the resynchronization data returns in the STSN response.

5 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is not the same as the receive sequence number in the resynchronization data. The receive indicator in the resynchronization data is off. The processing of the previous session, which left this condition, appears below. The sequence numbers shown are examples of those maintained by Series/1 SNA and by the host subsystem.

Previous Processing	Application Sequence Number	RI	PRI	Host Sequence Number
1. Both ends synchronized	10	off		10
2. Host sends a message	10	off		11
3. Link breaks before Series/1 SNA receives the message	10	off		11

In this case, the host sent an entire message to the Series/1 SNA application, but the link went down before the application processed the message. Note that the host sequence number was updated.

The "message from host lost" return code for this flow informs the application that a message that was not seen by the application was lost. The resynchronization data is updated to the host number.

The STSN response generated for this flow depends upon the STSN option selected by the host. The host always returns a result code of "ignore" (B'01') to the "set" option and handles the response to "set&test" as follows:

a The STSN result code is "mismatch."

b The receive sequence number from the resynchronization data returns in the STSN response.

6 The host selected the "set" or "set&test" option on the STSN. The sequence number on the STSN is not the same as the receive sequence number in the resynchronization data. The receive indicator is set in the resynchronization data. The processing of the previous session, which left this condition, appears below. The sequence numbers shown are examples of those maintained by Series/1 SNA and by the host subsystem.

Previous Processing	Application Sequence Number	RI	PRI	Host Sequence Number
1. Both ends synchronized	10	off		10
2. Host sends chain of three RUs.	10	off		13
3. Application issues NETGET	10	on		13
4. Link breaks before application gets all of chain.	10	on		13

In this case, the application issued one or more NETGET operations but did not receive the entire chain before the link broke. The return code on this NETINIT indicates "partially presented message from host lost" (4), informing the application that the host can send the message again and that the application has already seen part of the message. The return code warns of the possibility of a repeated message.

Series/1 SNA resets the RI indicator.

The STSN response generated for this flow depends upon the STSN option selected by the host. The host always returns a result code of "ignore" (B'01') to the "set" option and handles the response to "set&test" as follows:

a The STSN result code is "mismatch."

b The receive sequence number from the resynchronization data returns in the STSN response.

7 The host selected the "set" or "set&test" option on the STSN, with the sequence number of each flow being X'0000', indicating a host cold start. The receive indicator is reset, indicating that the application was not receiving a message when the host was cold started.

The receive sequence number is set to X'0000' to match the host. A "synchronized" return code is generated for this flow, indicating the host cold start. (This processing also occurs if no STSN is received.)

The STSN response generated for this flow depends upon the STSN option selected by the host. The host always returns a result code of "ignore" (B'01') to the "set" option and handles the response to "set&test" as follows:

a If the application chooses to initialize the contents of the resynchronization data, it uses the receive sequence number X'0000'; otherwise, it uses the receive sequence number from the resynchronization data.

b If the receive sequence number is X'0000', the result code is "match"; otherwise, it is "mismatch."

c The receive sequence number returns in the STSN response.

8 The host selected the "set" or "set&test" option on the STSN, with the sequence number of each flow being X'0000', indicating a host cold start. The receive indicator is set, meaning the application was receiving a message when the outage that caused the cold start occurred.

The return code to the application indicates that the message being received was lost due to the cold start of the host.

The receive sequence number is set to X'0000', and the receive indicator is reset. (This processing also occurs if no STSN is received.)

The STSN response generated for this flow depends upon the STSN option selected by the host. The host always returns a result code of "ignore" (B'01') to the "set" option and handles the response to "set&test" as follows:

a If the application chooses to initialize the contents of the resynchronization data, it uses the receive sequence number X'0000'; otherwise, it uses the receive sequence number from the resynchronization data.

b If the receive sequence number is X'0000', the result code is "match"; otherwise, it is "mismatch."

c The receive sequence number returns in the STSN response.

9 The host selected the "sense" option on the STSN. Series/1 SNA responds to the STSN with an "ok" result code (B'11') and includes the sequence number in the response. The host handles the sequence number as follows:

a If the application wants Series/1 SNA to initialize the resynchronization data set contents, a sequence number of X'0000' returns to the host.

b Otherwise, the receive sequence number from the resynchronization data is used.

STSN Processing for Sessions without Resynchronization Support

Figure 10-4 shows the processing performed for STSN on a session without resynchronization support. STSN processing is identical for PLU-to-SLU and SLU-to-PLU message flows. As seen in Figure 10-4, the resynchronization data is not processed.

RN	STSN Request	Sequence Numbers	Indicators	Series/1 SNA Action	Return Code	STSN Response
1	ignore (B'00')	N/A	N/A	none	SYNCHED	ignore (B'01')
2	set (B'01')	N/A	N/A	none; Resynch data not processed.	SYNCHED	ignore (B'01')
3	sense (B'10')	N/A	N/A	none; Resynch data not processed.	SYNCHED	invalid (B'10')
4	set&test (B'11')	N/A	N/A	none; Resynch data not processed.	SYNCHED	invalid (B'10')

Figure 10-4. STSN Processing for Sessions Without Resynchronization

Sessions without Resynchronization Support Explanation

The reference numbers in Figure 10-4 correspond to the numbered descriptions below.

1 The host selected the "ignore" option on the STSN for this message flow. Series/1 SNA takes no action on this STSN except to create a "synchronized" return code for this flow and an STSN result code of "ignore."

2 The host selected the "set" option on the STSN. Series/1 SNA does not process the resynchronization data. An STSN result code of "ignore" is generated.

3 The host selected the "sense" option on the STSN. Series/1 SNA does not process the resynchronization data. An STSN result code of "invalid" is generated, since sequence numbers are not maintained for this session.

4 The host selected the "set&test" option on the STSN. Series/1 SNA does not process the resynchronization data. An STSN result code of "invalid" is generated, since sequence numbers are not maintained for this session.

NETINIT Return Codes for STSN Processing

The entries in Figure 10-5 on page 10-22 show the NETINIT return codes returned to the application based on STSN processing. The return codes show the status of both the SLU-to-PLU and the PLU-to-SLU message flows.

The headings across the top of Figure 10-5 show the possible return codes for the PLU-to-SLU flow. The headings down the left side show the possible return codes for the SLU-to-PLU flow. The intersection of two return codes in Figure 10-5 is the resulting NETINIT return code.

In Figure 10-5 the following terms are used:

Term **Meaning**

000 – 081 Nonerror combination return codes, indicating the status of both message flows.

The positive return codes from NETINIT have bit-significant values with the following meaning:

Message flow from host (PLU-to-SLU)

```
.... .... .... ....  Message flow from host synchronized.
.... .... .... ...1  Message flow from host cold-started.
.... .... .... ..1.  Message from host lost.
.... .... .... .1..  Partial message from host lost.
```

Message flow to host (SLU-to-PLU)

```
.... .... .... ....  Message flow to host synchronized.
.... .... ...1 ....  Message flow to host cold-started.
.... .... ..1. ....  Message to host lost.
.... .... .1.. ....  Message to host possibly lost.
```

--- This combination cannot occur.

-31 An error was encountered such that the application and the host cannot be synchronized.

PLU-to-SLU

SLU-to-PLU

	SYNC	Host Cold Start	Message From Host Lost	Partial Message From Host Lost	Message From Host Lost + Host Cold Start
SYNCHED	0	---	2	4	---
Host Cold Start	---	17	---	---	19
Message to Host Lost	32	---	-31	-31	---
Possible Message to Host lost + Host cold Start	---	81	---	---	---
Message to Host lost + Host cold Start	---	49	---	---	---
STSN error	-31	-31	-31	-31	-31

B0773002

Figure 10-5. NETINIT Return Codes for STSN Processing

Examples

```
* EXAMPLE OF STARTING A SESSION WITHOUT RESYNCHRONIZATION
*
* * NETINIT FOR ANY LU ON PU #3
*
* * NETINIT SHOULD FAIL IF THE LU-SSCP SESSION IS NOT ACTIVE
*
* * PRIORITY FOR ALL OUTBOUND MESSAGES IS 80 WHILE THE LU IS IN SESSION
*
       NETINIT LU=NETLU,          ANY LU ON PU 3                          C
               HOSTID=SNAID,      SPECIFY WHICH HOST APPL                 C
               ACQUIRE=NO,        LET THE HOST START THE SESSION          C
               RESYNC=NO,         MESSAGE RESYNCHRONIZATION NOT USED      C
               MSGDATA=MDATA,     PASS BACK THE SEQUENCE NUMBERS          C
               LUSWAIT=NO,        FAIL IF LU-SSCP SESSION IS INACTIVE     C
               MSGPRIO=MPRIO      MESSAGE PRIORITY IS 80
            •
            •
            •
NETLU    DATA  X'0300'            PU NUMBER 3, ANY LU
MDATA    DATA  6F'0'              MSGDATA AREA
SNAID    NETHOST ISAPPID=JES2,ISMODE=RMT26  TALK TO JES2 APPL ON HOST
MPRIO    DATA  F'80'              MESSAGE PRIORITY = 80
```

Figure 10-6. Starting a Session without Resynchronization

```
* EXAMPLE OF STARTING A SESSION WITH RESYNCHRONIZATION DATA TO STORAGE
*
* * THE LU IS NUMBER 1, ON PU 1, SPECIFIED AT LOCATION NETLU
*
* * NETINIT SHOULD WAIT IF THE LU-SSCP SESSION IS NOT ACTIVE
*
* * PRIORITY FOR ALL OUTBOUND MESSAGES IS 40 WHILE THE LU IS IN SESSION
*
       NETINIT LU=NETLU,          LU #1 on PU #1                          C
               HOSTID=SNAID,      SPECIFY WHICH HOST APPL                 C
               ACQUIRE=NO,        LET THE HOST START THE SESSION          C
               RESYNC=INIT,       START MSG RESYNCH                       C
               MSGDATA=MDATA,     RESYNCH DATA IN STORAGE                 C
               LUSWAIT=YES,       WAIT FOR LU-SSCP SESSION                C
               MSGPRIO=MPRIO,     MESSAGE PRIORITY IS 40                  C
               RTYPE=STG          SAVE SYNCHRONIZATION DATA IN STORAGE
            •
            •
            •
NETLU    DATA X'0101'             PU #1 LU #1
MDATA    DATA 6F'0'               MSGDATA AREA
SNAID    NETHOST ISAPPID=JES2,ISMODE=RMT26  TALK TO JES2 APPL ON HOST
MPRIO    DATA F'40'               MESSAGE PRIORITY IS 40
```

Figure 10-7. Starting a Session with Resynchronization

Chapter 11. Extended Error Information

If you established a session specifying the NETINIT ERRCODE= parameter, Series/1 SNA support returns more error information to your application (when appropriate) after an "SNA system error" (−14) return code and *other negative* return codes. SNA support places the function that failed and its return code in the area specified by the ERRCODE parameter. This area is meaningful only when a negative return code (other than −1) returns to your application. If the return code is not "system error" (−14), the ERRCODE area may or may not have been filled. The failing function is identified by a 1-byte hexadecimal code. These codes are listed in Figure 11-1, along with the format of the extended error code area.

Displacement in bytes Dec	Hex	Field Length	Description
0	00	1	**SNA Instruction:** X'00' NETINIT X'01' NETPUT X'02' NETGET X'03' NETCTL X'05' NETTERM
1	01	1	**EDX SNA Function:** X'01' NETOPEN X'02' NETRECV X'03' NETSEND X'04' NETCLOSE X'05' NETBIND X'06' NETUBND X'08' BIND event post code X'0A' READ X'0B' WRITE X'0C' Session termination
2	02	2	**EDX Return Code**

Figure 11-1. Hexadecimal Codes and Extended Error Code Area Format

Extended Error Return Code Descriptions

This section describes the extended error information returned to your application from a failing EDX function. The return codes from the following functions are described:

NETOPEN
NETRECV
NETSEND
NETCLOSE
NETBIND
NETUBND
BIND event post codes
READ
WRITE
Session termination.

NETOPEN Return Codes

NETOPEN Return Code	Condition
1	UNBIND received.
-9	Session terminated.
-10	Nonsequence procedure error message received.
-11	Minus response to INIT-SELF received.
-13	BIND has not been received yet.
-14	BIND request received does not compare with user-specified BIND during extended BIND checking.
-15	Received BIND with an invalid parameter.
-17	Network pending deactivation.
-18	Session aborted.
-57	No buffer available for INIT-SELF message.
-58	INIT-SELF message greater than buffer size.
-68	SSCP ID in the PUS not the same as user-supplied SSCP ID.
-71	UNBIND received.
-79	BIND buffer length does not equal user-specified BIND length.
-82	No LU-SSCP session (returned when NETOPEN waits on a LU-SSCP session activation and some condition inhibits the activation).
-85	HOLDLU specified but session not closed with HOLD=YES.
-86	HOLDLU specifies an invalid LU number.

Figure 11-2 (Part 1 of 2). NETOPEN Return Codes

NETOPEN Return Code	Condition
-87	LU-SSCP session not active and you did not want to wait for the session to become active.
-89	The I-Frame sizes for EDX SNA and EDX Primary SNA do not match.
-90	The primary session specified by the PACB value on the NETINIT is not a passthru session.
-91	The primary session specified by the PACB value on the NETINIT is not valid.
-92	The EDX Primary SNA program, $PSNA, is not loaded.
-93	The Primary SNA network is currently activating.
-94	The Primary SNA network is deactivating.
-95	The Primary SNA network is inactive.
-96	The EDX Primary SNA program, $PSNA, is in the process of unloading and is, therefore, not usable.
-97	The Primary SNA passthru session has been terminated or was never open.
-200	Primary SNA not generated into system and you requested passthru.

Figure 11-2 (Part 2 of 2). NETOPEN Return Codes

NETRECV Return Codes

NETRECV Return Code	Condition
1	Successful and message available to be received.
2	Message truncated.
3	Message truncated and message available to be received.
-9	Session terminated.
-10	CLEAR received.
-12	No messages available.

Figure 11-3. NETRECV Return Codes

NETSEND Return Codes

NETSEND Return Code	Condition
-9	Session terminated.
-10	CLEAR received.
-12	Format indicator is on, on this request, and the session is bound no FM headers.
-13	Session bound RQN requests only and this request is not RQN.
-14	This request carries a BB or EB and not BC.
-15	Trying to send a request when the brackets manager is not one of the following states: BETB, PEND.BB, INB PEND.BETB.ECS PEND.BETB.PURGE.S PEND.BETB.S
-16	Trying to send a request when not on a "send" state.
-17	QRI=OFF on this request, when sending a set of QRIs.
-18	Secondary session quiesced due to a receipt of QEC or SHUTD.
-19	Trying to send BC while the chain receive manager is in the INCHAIN state and this request in not a cancel.
-20	QC while not in PEND.ACT.QC.
-21	CANCEL required. Half session partner in purge state.
-22	Trying to send a RQD request that does not carry EC.
-24	Data traffic not active.
-25	Negative response to SHUTD not allowed.
-26	No normal flow response due.
-28	Invalid RU code for normal/expedited send request.
-29	The form of response is invalid, the response is not RQN, RQE or RQD.
-31	Response required to an EXR.
-32	Positive response to a RQE request not allowed.
-33	Invalid data length specified.
-34	Maximum RU length exceeded.
-35	Response to an outstanding RQD request has not been received.

Figure 11-4 (Part 1 of 3). NETSEND Return Codes

NETSEND Return Code	Condition
-36	Response to an outstanding expedited request has not been received.
-37	Negative response to QEC not allowed.
-38	Negative response to RELQ not allowed.
-39	State error in sending SHUTC while the SHUTD.RCV FSM not on PEND.ACT.SHUTC.
-40	No response due to a normal request with this sequence number.
-41	Invalid RU for NRM response.
-42	No response due to an expedited request.
-43	No response due to an expedited request with this ID.
-44	No response due to this expedited RU.
-45	Trying to send a DFC command with EB while BSM is in the BETB state.
-46	Trying to send a FMD request with no BB while BSM is in the BETB state.
-47	Trying to send a FMD request with BB while BSM is in the INB state.
-48	Error in trying to send RTR while BSM is in the INB state.
-49	Trying to send a FMD request or CANCEL or RTR or any DFC command with EB while BSM is in PEND.BB state.
-50	CSI set on by this request and the session is bound, no alternate code.
-51	Secondary LU not allowed to send EB.
-52	Session bound RQD requests only, and this request is an RQN request.
-53	Session bound RQD requests only, and this request is an RQE request with EC.
-54	Session bound RQD or RQE requests only and this request is an RQN request.
-55	Session bound RQE requests only and this request is not an RQE request.
-56	Session bound no brackets, and this request has BB specified.
-57	Chain receive manager is in between chains state, and this request does not carry BC.

Figure 11-4 (Part 2 of 3). NETSEND Return Codes

NETSEND Return Code	Condition
-58	Chain receive manager is in between chains state and this is a CANCEL request.
-59	The half-duplex manager is in the contention ERP state and this request is not LUSTAT.
-60	The half-duplex manager is in the contention ERP state, and this request is LUSTAT with EB and/or CD.
-62	Invalid request for the FM profile for which the session was bound.
-63	Sending QEC when QECS.FSM not in reset state or active state.
-64	Invalid request for the TS profile for which the session was bound.
-66	CDDCLEAR failed.
-67	This change of direction request does not carry EC.
-68	Invalid parameter list, does not specify a valid SNA function.
-69	Error in trying to send RTR when session is bound without brackets.
-70	Negative response to STSN is not allowed.
-71	The session was bound single RU chains from the secondary, and this request is not.
-72	This function management header request does not carry BC.
-73	Response to an outstanding RQR request has not been received.
-74	Violation of the session quiesced protocol. QECR FSM or SHUTD FSM are in the P.ACT.QC/P.ACT.SHUTC state.
-75	A request without both BB and EB indicators was sent on a duplex session supporting brackets.

Figure 11-4 (Part 3 of 3). NETSEND Return Codes

NETCLOSE Return Codes

NETCLOSE Return Code	Condition
-51	Invalid session ID.
-301	No buffer available to send TERMSELF.
-306	HOLD=YES but no UNBIND HOLD.
-307	HOLD=YES but already held.

Figure 11-5. NETCLOSE Return Codes

NETBIND Return Codes

NETBIND Return Code	Condition
-9	Session terminated.
-10	Clear received.
-13	BIND has not been received yet.
-15	Received BIND with invalid parameter.
-16	Network aborted.
-18	BIND FSM state not reset when checking BIND.
-19	BIND FSM state not PEND.ACT when sending response.

Figure 11-6. NETBIND Return Codes

NETUBND Return Codes

NETUBND Return Code	Condition
-9	Session terminated.
-10	Clear received.
-12	TERMSELF already sent.
-13	No buffer available for TERMSELF.

Figure 11-7. NETUBND Return Codes

BIND Event Post Codes

BIND Post Code	Condition
1	UNBIND received.
2	UNBIND HOLD received.
-9	Session terminated.
-10	Non-sequenced procedure error received.
-11	Minus response to INIT-SELF or TERMSELF received.
-12	A primary SNA LU was not available within the range specified for the primary SNA PU. The BIND request was rejected. (PASSTHRU ONLY)

Figure 11-8. BIND Event Post Codes

READ Return Codes

READ Return Code	Condition
-1	Successful completion.
1	I/O error and no device status present (this code may be caused by the I/O area starting at an odd byte address).
2	I/O error trying to read device status.
3	I/O error retry count exhausted.
4	Read device status I/O instruction error.
5	Unrecoverable I/O error.
6	Error on issuing I/O instruction for normal I/O.
7	A "no record found" condition occurred, a seek for an alternate sector was performed, and another "no record found" occurred, for example, no alternate is assigned.
8	A system error occurred while processing an I/O request for a 1024-byte sector diskette.
9	Device was offline when I/O was requested.
10	Record number out of range of data set—may be an end-of-file (data set) condition.
11	Data set not open or device marked unusable when I/O was requested.
12	DSCB was not open; DDB address = 0.
13	If extended deleted record support was requested ($DCSBFLG bit 3 on), the referenced sector was not formatted at 128 bytes/sector or the request was for more than one 256-byte sector. If extended deleted record support was not requested ($DSCBFLG bit 3 off), a deleted sector was encountered during I/O.
14	The first sector of the requested record was deleted.
15	The second sector of the requested record was deleted.
16	The first and second sectors of the requested record were deleted.

Figure 11-9. READ Return Codes

WRITE Return Codes

WRITE Return Code	Condition
-1	Successful completion.
1	I/O error and no device status present (this code may be caused by the I/O area starting at an odd byte address).
2	I/O error trying to read device status.
3	I/O error retry count exhausted.
4	Read device status I/O instruction error.
5	Unrecoverable I/O error.
6	Error on issuing I/O instruction for normal I/O.
7	A "no record found" condition occurred, a seek for an alternate sector was performed, and another "no record found" occurred, for example, no alternate is assigned.
8	A system error occurred while processing an I/O request for a 1024-byte sector diskette.
9	Device was offline when I/O was requested.
10	Record number out of range of data set—may be an end-of-file (data set) condition.
11	Data set not open or device marked unusable when I/O was requested.
12	DSCB was not open; DDB address = 0.
13	If extended deleted record support was requested ($DCSBFLG bit 3 on), the referenced sector was not formatted at 128 bytes/sector or the request was for more than one 256-byte sector. If extended deleted record support was not requested ($DSCBFLG bit 3 off), a deleted sector was encountered during I/O.
14	The first sector of the requested record was deleted.
15	The second sector of the requested record was deleted.
16	The first and second sectors of the requested record were deleted.

Figure 11-10. WRITE Return Codes

Session Termination Return Codes

Return Code	Condition
0	Session active.
1	BIND not received.
2	UNBIND received.
3	BIND response not sent.
4	CLOSE in progress.
5	ACTLU received.
6	DACTLU received.
7	Buffer count exceeded.
8	SDLC failure (hard I/O error).
9	Request discontact failure.
10	DISC received.
11	ACTPU received.
12	DACTPU received.
13	Negative response to request disconnect received.
14	Network aborted.
15	UNBIND HOLD received.

Figure 11-11. Session Termination Return Codes

Attention Event Post Codes for Passthru

Post Code	Condition
1	UNBIND sent by the remote primary LU to the remote secondary LU.
-1	BIND sent by the remote primary LU to the remote secondary LU.
-9	Passthru session terminated; issue NETTERM.
-14	No stack available to send message through the passthru link. Primary SNA PU in the Series/1 may be deactivating; issue NETTERM.

Figure 11-12. Attention Event Post Codes for Passthru

Chapter 12. SNA Base Support

SNA base support in Series/1 is a subset of the total SNA architecture. The support provides services to establish, control, and terminate sessions between multiple Series/1 programs and System/370 subsystems or user programs within an SNA network. The support also provides services to transfer data and control information between the programs. These services are provided for by use of SNA protocols.

The network is defined during the system generation process. The network communication process consists of:

- Network activation
- Sessions established
- Message exchange
- Session termination
- Network deactivation.

Defining the Network

If you include SNA in the system, you must include EXIO and TIMER support in your supervisor during system generation.

During network generation, you can supply information on the characteristics of each PU, each SDLC communications line and device, and each 3705 connecting station. Specify this information on the SNAPU and SNALU configuration statements, which build the control blocks and tables for your SNA network.

See Chapter 2, "Installing and Defining the SNA Network for Use with SDLC" for more information on the SNAPU and SNALU configuration statements. These configuration statements perform the following:

SNAPU Generates a PU control block (SDLC line and station protocol)

SNALU Generates an LU control block (number of the LU and session buffers) for each LU associated with the PU.

See Chapter 3, "Installing and Defining the SNA Network for Use with Shared SDLC" for more information on the SNAPU and SNALU configuration statements. These configuration statements perform the following:

SNAPU Generates a PU control block only

SNALU Generates an LU control block (number of the LU and session buffers) for each LU associated with the PU.

Session Activation and Deactivation

Session activation is the process of dynamically establishing an LU-to-LU communications path for subsequent data and control command exchange. Session deactivation is the process of terminating the LU-to-LU connection.

In Series/1 SNA support, an LU session is a connection between a primary LU (a System/370 user program or subsystem) and a secondary LU (a Series/1 program). You can initiate a request for session activation or deactivation by either a primary or secondary LU; however, only the primary LU can issue the command that establishes the session (BIND).

SNA provides communication support for the following specific LU type 0 definitions:

IMS/VS IBM 3790 Logical Unit type P

CICS/VS IBM 3790 Full Function Logical Unit

CICS/VS IBM 3650 Pipeline Logical Unit

Message Exchange

The message exchange facility allows the exchange of data between a Series/1 program and its System/370 session partner. The data may consist of Series/1 or System/370 application data, SNA commands that control the flow of data through the network, or Series/1 or System/370 responses to requests.

In SNA, the request/response header is the control field that specifies the type of data being transmitted and the control information associated with the data.

Chapter 13. SNA Protocols

SNA defines many protocols (or procedures) for communicating within an SNA network, including:

- Function management profiles
- Transmission subsystem profiles
- BIND checking
- BIND parameters
- Correlation tables
- Brackets
- Chains
- Half-duplex flip-flop
- Duplex
- Pacing
- SNA commands.

Function Management Profiles

Function management (FM) profiles specify which data flow control facilities to use during a session. For Series/1 SNA support, you can use function management profiles 3 and 4.

Profile 3 is a subset of profile 4. The differences between profiles 3 and 4 are:

- Profile 3 provides that the secondary LU can send only the LUSTAT command, whereas profile 4 provides that the secondary LU can send or receive LUSTAT.
- Profile 4 provides quiesce functions.

Figure 13-1 provides a listing of the Series/1 SNA commands supported by profiles 3 and 4 and identifies whether the secondary LU can send and/or receive each command.

Profile 3	Profile 4	Command	Secondary LU Send	Secondary LU Receive
X	X	Bid (BID)		X
X	X	Cancel (CANCEL)	X	X
X	X	Chase (CHASE)	X	X
X	X	Ready to Receive (RTR)[1]	X	

Figure 13-1 (Part 1 of 2). SNA Commands Supported by FM Profiles 3 and 4

[1] Allowed only if you use brackets.

Profile 3	Profile 4	Command	Secondary LU Send	Secondary LU Receive
X	X	Request Shutdown (RSHUTD)	X	
X	X	Shutdown (SHUTD)		X
X	X	Shutdown Complete (SHUTC)	X	
X	X	Signal (SIG)	X	X
X		Logical Unit Status (LUSTAT)	X	
	X	Logical Unit Status (LUSTAT)	X	X
	X	Quiesce at End of Chain (QEC)	X	X
	X	Quiesce Complete (QC)	X	X
	X	Release Quiesce (RELQ)	X	X

Figure 13-1 (Part 2 of 2). SNA Commands Supported by FM Profiles 3 and 4

The session restrictions for primary and secondary LUs enforced by profiles 3 and 4 include:

- You can transmit multiple request units before requesting a definite response (immediate request mode).
- Responses must return in the same order as the LU receives associated requests (immediate response mode).
- SNA does not allow network services commands.

In SNA, the request/response header is the control field that specifies the type of data being transmitted and the control information associated with the data. Function management profiles 3 and 4 permit you to set the following indicators in the request/response header:

- Request/response unit category (in RU category field)
- Function management header (in FI field)
- Chaining (in BCI and ECI fields)
- Response type request (in DR1, DR2, and ERI fields)
- Queued response indicator (in QRI field)
- Brackets (in BBI and EBI fields)
- Change of direction (in CDI field)
- Alternate code (in CSI field)
- Positive/negative response indicator (in RTI field).

Transmission Subsystem Profiles

Transmission subsystem (TS) profiles specify which transmission subsystem facilities to use during a session. For Series/1 SNA support, you can use transmission subsystem profiles 3 and 4.

Profile 3 is a subset of profile 4. The difference between profiles 3 and 4 is the addition of the STSN command for profile 4.

Figure 13-2 provides a listing of the Series/1 SNA commands supported by profiles 3 and 4 and identifies whether the secondary LU or PU can send and/or receive each command.

Profile 3	Profile 4	Command	Secondary LU/PU Send	Secondary LU/PU Receive
X	X	Activate Logical Unit (ACTLU)[2]		X
X	X	Activate Physical Unit (ACTPU)[2]		X
X	X	Deactivate Logical Unit (DACTLU)[2]		X
X	X	Deactivate Physical Unit (DACTPU)[2]		X
X	X	Bind Session (BIND)		X
X	X	Unbind Session (UNBIND)[2]	X	X
X	X	Start Data Traffic (SDT)		X
X	X	Clear (CLEAR)[2]		X
X	X	NS Procedure Error (NSPE)[2]		X
X	X	Initiate Self (INIT-SELF)[2]	X	
X	X	Terminate Self (TERM-SELF)[2]	X	
X	X	Request Discontact (REQDISCONT)	X	
	X	Set and Test Sequence Numbers (STSN)		X

Figure 13-2. SNA Commands Supported by TS Profiles 3 and 4

In Figure 13-2, commands NSPE, INIT-SELF, TERM-SELF, and REQDISCONT are network services commands. All of the other commands are session control commands.

[2] Processed for you by Series/1 SNA support.

BIND Checking

A primary LU sends a BIND command to a secondary LU to activate a session between the LUs. The BIND command contains the session parameters that a secondary LU uses to determine whether or not a session can be activated.

When the system receives the BIND, it checks the fields to ensure compatibility with Series/1 SNA support. This check is called *basic BIND checking*. If any of the checks fail, a negative response goes to the BIND and the session is not activated.

The bind parameters indicated with an asterisk (*) in Figure 13-3 are checked during basic BIND checking.

BIND Parameters

The following description of the BIND command shows all the BIND parameters that are valid with Series/1 SNA support:

Byte	Bit	Description	Value
0	0-7	BIND Command	X'31'
1*	0-3	**Format:** Zero	X'0'
1*	4-7	**Type:** Cold	X'1'
2*	0-7	**Function management profile indicator:** Profile 3 Profile 4	 X'03' X'04'
3*	0-7	**Transmission subsystem profile indicator:** Profile 3 Profile 4	 X'03' X'04'
4	0	**Primary chaining indicator:** Single element chains only Multiple element chains	 B'0' B'1'
4*	1	**Primary request control mode indicator:** Immediate request mode	 B'0'
4	2-3	**Primary chain response protocol indicator:** No response Exception response only Definite response only Exception or definite response	 B'00' B'01' B'10' B'11'
4	4-5	Reserved	B'00'

Figure 13-3 (Part 1 of 4). BIND Parameters

Byte	Bit	Description	Value
4	6	**Primary compression indicator:** No compression Compression	 B'0' B'1'
4	7	**Primary send end bracket indicator:** Will not send end bracket Can send end bracket	 B'0' B'1'
5	0	**Secondary chaining indicator:** Single element chains only Multiple element chains	 B'0' B'1'
5*	1	**Secondary request control mode indicator:** Immediate request mode	 B'0'
5	2-3	**Secondary chain response protocol indicator:** No response Exception response only Definite response only Exception or definite response	 B'00' B'01' B'10' B'11'
5	4-5	Reserved	B'00'
5	6	**Secondary compression indicator:** No compression Compression	 B'0' B'1'
5	7	**Secondary send end bracket indicator:** Does not send end bracket Can send end bracket	 B'0' B'1'
6	0	Reserved	B'0'
6	1	**Function management header indicator:** Header not allowed Header allowed	 B'0' B'1'
6	2	**Bracket usage and reset state indicator:** Do not use brackets if neither primary nor secondary send EB. Use brackets if either primary or secondary can send EB and reset state is in brackets (INB) Use brackets and reset state is between brackets (BETB)	 B'0' B'1'

Figure 13-3 (Part 2 of 4). BIND Parameters

Byte	Bit	Description	Value
6	3	**Bracket termination rule indicator:** Unconditional (rule 2) Conditional (rule 1)	 B'0' B'1'
6	4	**Alternate code indicator:** No alternate code Alternate code allowed	 B'0' B'1'
6	5-7	Reserved	B'000'
7*	0-1	**Normal flow send/receive mode indicator:** Half-duplex flip-flop	B'10'
7*	2	**Recovery responsibility indicator:** Primary error recovery procedure	B'0'
7*	3	**Brackets first speaker indicator:** Secondary is first speaker	B'0'
7	4-6	Reserved	B'000'
7*	7	**Contention resolution indicator:** Secondary sends first	B'0'
8	0-1	Reserved	B'00'
8	2-7	**Secondary connection point manager send pacing count**	
9	0-1	Reserved	B'00'
9	2-7	**Secondary connection point manager receive pacing count**	
10	0-7	**Maximum request unit size sent by secondary**	
11	0-7	**Maximum request unit size sent by primary**	
12	0-1	Reserved	B'00'
12	2-7	**Primary connection point manager send pacing count**	
13	0-1	Reserved	B'00'
13	2-7	**Primary connection point manager receive pacing count**	
14	0	**Logical unit profile format:** Basic	B'0'
14	1-7	**Logical unit type**	
15-25		**Logical unit characteristics**	

Figure 13-3 (Part 3 of 4). BIND Parameters

Byte	Bit	Description	Value
26	0-1	**Private and end-user encryption indicator:** No private or end-user encryption Private encryption only End-user encryption only	 B'00' B'01' B'10'
26*	2-3	**Session-level encryption indicator:** No session-level encryption	 B'00'
26	4-7	**End-user encryption indicator:** No end-user encryption specified. If this value is specified, fields 27-K are omitted. End-user encryption specified	 X'0' X'9'
27	0-1	**Session encryption key enciphered indicator:** Session encryption key enciphered under SLU master encryption crypto using a seed value of zero.	 B'00'
27	2-4	Reserved	B'000'
27	5-7	**Block chaining indicator:** Block chaining with seed and cipher text feedback using the data encryption standard (DES) algorithm	 B'000'
28-K		**Session encryption key**	
(K+1)	0-7	**Length of primary LU name:** If you specify 0 for this value, fields (K+2)-M are omitted.	
(K+2)-M		**Primary LU name**	
(M+1)	0-7	**Length of user data:** If you specify 0 for this value, fields (M+2)-N are omitted.	
(M+2)-N		**User data**	

Figure 13-3 (Part 4 of 4). BIND Parameters

Correlation Tables

Use correlation tables to prevent invalid responses from being received and to ensure that only valid responses are sent. For example, you can use correlation tables to ensure that:

- If you send a chain with sequence numbers ranging from 5 to 10, only responses with sequence numbers from 5 to 10 are received.
- If you attempt to send a positive response to an exception response chain, the positive response is rejected.

- If you attempt to send a response with a sequence number of 15 but have received a chain with sequence numbers ranging from 9 to 14, the response with sequence number 15 is rejected.

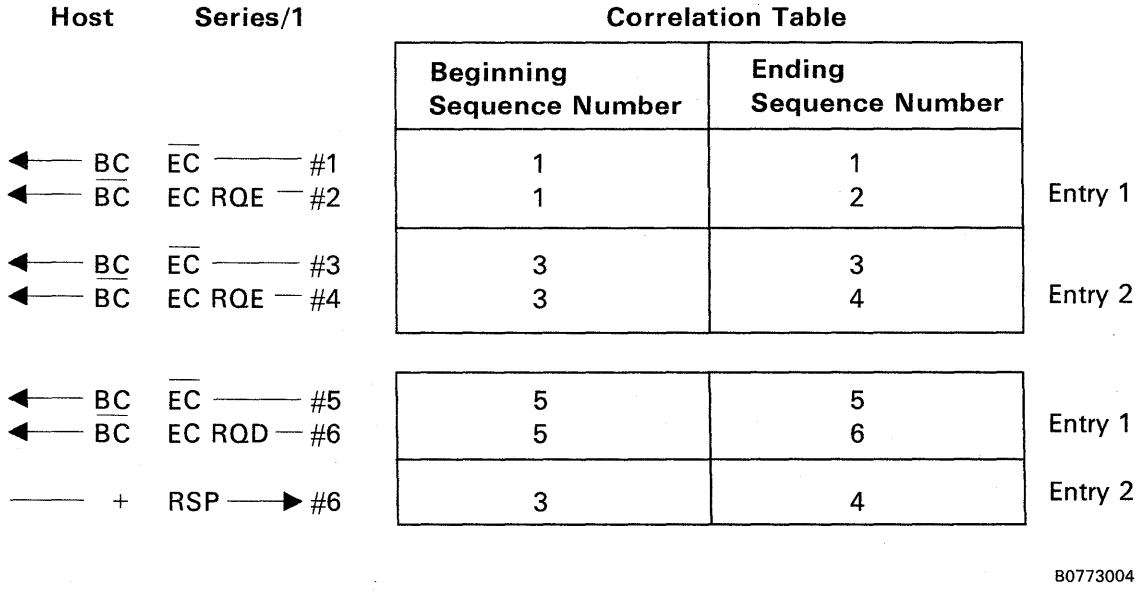
SNA maintains an entry in a correlation table for each request chain that is sent or received by the Series/1 program. Following is a diagram of an entry.

Byte		
0	Request/response header Byte 0	Request/response header Byte 1
2	Request/response header Byte 2	RU code
4	Beginning sequence number	
6	Ending sequence number	

B0773003

Figure 13-4. Correlation Table Entry

Each entry in the table contains the RHDATA area information that reflects the current element of the chain that was sent or received. SNA updates this information to reflect that of the last (or current) element sent or received. Each entry in the table also contains the sequence number of the first request in the chain and the sequence number of the last (or current) request in the chain. When the first element of a chain is sent or received, an entry in the correlation table is built for the chain, and the sequence number corresponding to that first request is placed in the entry. Every time an element of a chain is sent or received, the entry for the chain is updated to reflect the last (or current) sequence number corresponding to that element. The following illustration describes this action when CTE=2.



B0773004

Figure 13-5. Correlation Table Sequence Number Example (CTE=2). Note that the final "Entry 2" contains residual values from the previous "Entry 2".

The following list provides the acronyms used in the above correlation table with a brief explanation of their meaning.

BC	Begin chain
$\overline{\text{BC}}$	Not begin chain
EC	End chain
$\overline{\text{EC}}$	Not end chain
RQD	Request definite response
RQE	Request exception response

Series/1 SNA support maintains four correlation tables:

- For normal flow requests sent
- For normal flow requests received
- For expedited flow requests sent
- For expedited flow requests received.

Normal Flow Requests Sent

The correlation table maintained for normal flow requests sent consists of a variable number of entries as specified by the CTE parameter of the SNALU configuration statement. When your application issues NETPUT to send a request, it builds a table entry.

The table correlates requests sent with responses received:

- If you send a definite response request, SNA makes no additional entries in the table and rejects all NETPUT requests in the normal flow until your application receives a response.
- When your application receives a response for an entry, SNA eliminates that entry and all previous entries. (SNA assumes that the System/370 user accepted the previous entries.)
- If the correlation table is full when a request is sent, the table "wraps" and overlays the oldest entry. (SNA assumes that the System/370 user accepted the overlaid entry.) To ensure that the table does not wrap, you should send a request requiring a definite response before the table wraps.

The CTE value to be specified depends entirely on your application. For example, if the primary LU requires (as a BIND specification) that it receive only definite response request chains, you need only one outbound correlation table entry. In the most extreme case, the highest CTE value that you can specify should be one more than the number of exception-response request chains that transmit before sending a definite-response request. In all cases, the CTE value specified should depend only on the environment in which you use SNA support.

Normal Flow Requests Received

The correlation table for normal flow requests received consists of only one entry. SNA builds an entry when your application issues a NETGET instruction and when the request it receives carries the begin chain (BC) indicator. SNA updates the entry every time the application reissues the NETGET instruction and receives a request. The table correlates requests received with responses sent:

- If you receive an exception-response request and you wish to respond negatively, you must do so before receiving the next element in the chain. Also, if you do not respond negatively to the last element of a chain before receiving the first element of the next chain, you can no longer respond to the previous chain.
- If you receive a definite response request, you must respond, and no additional requests are sent to you until you respond. Therefore, in this situation, the entry can never be erroneously overlaid.

Expedited Flow Requests Sent and Received

The correlation tables for expedited flow requests sent and received each consist of only one entry, as described for normal flow requests received.

Chains

A chain is a set of one or more requests that are transmitted through a network as one recognizable entity in the context of error recovery. A response to any request in the chain either accepts or rejects the entire chain.

Each chain contains the following properties:

- All elements in the chain are requests.
- All requests in the chain belong to the same flow.
- The first request in the chain is marked BC (begin chain).
- The last request in the chain is marked EC (end chain).
- Each middle request in the chain is marked not BC, not EC.
- A chain having only one request is marked BC, EC.
- All expedited requests are single-element chains (marked BC, EC).
- The only normal flow data flow control command that can be sent while sending a normal flow multiple-request chain is CANCEL, which terminates the chain.

The sender of a chain must set the "form of response requested" bits in each request of the chain. The sender can send only chains of the following types:

No-response
Each request in the chain is marked no-response.

Exception-response
Each request in the chain is marked exception-response.

Definite-response
The last request in the chain is marked definite-response

All others
Marked exception-response.

The receiver of a chain must examine the "form of response requested" bits only in the last request in a chain or in a request in error. When the chain response BIND protocol parameter indicates that only definite-response chains, exception-response chains, or no-response chains will be received, the receiver need not examine and can assume without checking the "form of response required" bits.

If the chain sender is notified of an error in a chain being sent, the sender must reset the chain finite state machines by issuing either an EC function management data request, a CANCEL command (which carries EC), or a higher-level reset command (for example, CLEAR or UNBIND). Finite state machines (FSM) process a defined number of states (memory) and a set of rules, whereby its response (state changes and output sequences) to input sequences is well defined. When a Series/1 LU is the chain sender and is notified of an error in the chain, the Series/1 must use the CANCEL command to terminate the chain. (The Series/1 cannot terminate the chain by sending EC.)

When your Series/1 program sends a message that violates any of the chaining protocols, you receive a negative return code. When your Series/1 program receives a message about violation of the protocols, you receive an exception response.

Brackets

A bracket is a sequence of one or more normal flow request chains and their responses that are exchanged in either or both directions between a System/370 LU and a Series/1 LU. Each bracket essentially represents an uninterruptible unit of work.

Bracket protocol involves bracket initiation and termination rules and provides a method of communication in which a new bracket does not start until the bracket in progress completes. Following is a description of bracket protocol as supported by Series/1 SNA.

Bracket Initiation Rules

Series/1 support provides for the following definitions:

- The primary LU (the System/370 LU) is the bidder. The bidder is the LU defined at session initiation as having to request and receive permission from the secondary LU to begin a bracket.
- The secondary LU (the Series/1 LU) is the first speaker. The first speaker is the LU defined at session initiation as having the ability to begin a bracket without requesting permission from the primary LU and winning contention if both LUs attempt to begin a bracket simultaneously.

The primary LU can request permission to initiate a bracket by the BID normal-flow data flow control request. The primary LU must issue BID while its bracket state manager is in the between brackets state in order to be accepted by the secondary LU. Processing of the BID command through data flow control results in the request being converted into an exception request when the brackets finite state machine is in a state other than between brackets state. In this case, you must send a negative response indicating that the primary LU cannot begin a bracket. When the brackets finite state machine is in between brackets state, you can send a positive response indicating that the primary LU can begin a bracket, or you can send a negative response.

The secondary LU can initiate a bracket by a function management data request with the begin bracket (BB) indicator on.

Ready to Receive (RTR) is a normal-flow data flow control command that the secondary LU can send to indicate to the primary LU (type 0) that it can initiate a bracket.

The primary LU can also attempt to begin a bracket by sending a definite response function management data request that has a begin bracket (BB) indicator on. The secondary LU must respond either positively to begin the bracket or negatively to reject the request to initiate the bracket.

Bracket Indicator Rules

The following rules must be observed when using brackets in an LU-LU session:

- You can set BB only on the first (or only) request of a chain. You can set EB only on the first (or only) request of a chain. You can set BB and EB together only on the first (or only) request of a chain.
- You can set EB on CANCEL, LUSTAT, or QC commands.
- You can set BB or EB by either LU, unless restricted by a BIND parameter or a private end-user protocol.
- You can set EB on any normal flow data flow control requests except BID or RTR. You cannot set BB on any data flow control requests.
- You can set BB or EB on function management data requests. When the bidder sends a function management data request with BB, the request must be a definite response request. When the bidder sends a function management data request with both BB and EB, the request may be a definite, exception, or no-response request.
- After sending a positive response to BID or receiving a positive response to RTR, the first speaker must wait for the bidder to send a function management data request with BB.
- You cannot set either BB or EB on responses or expedited requests.

Bracket Termination Rules

Bracket protocol allows the termination of the bracket in two ways: conditionally by termination rule 1, and unconditionally by termination rule 2.

If a chain contains EB but not BB, the form of response requested controls conditional bracket termination rule 1.

- Definite response — If the chain requests a definite response, the bracket terminates only after processing of a positive response.

  If a negative response to the last request in the chain is processed, the bracket continues.

  If a negative response to a request other than the last request is processed, the sender of the chain has the option of terminating or continuing the bracket:

  - The sender can terminate the bracket by (1) sending CANCEL with EB or (2) ending the chain with a request specifying exception response or no-response.

  - The sender can continue the bracket by (1) sending CANCEL without EB or (2) ending the chain with a request specifying definite response.

- Exception response — If the chain requests an exception response, the bracket terminates unconditionally after processing of the last request of the chain that has EB in its first request.
- No-response — If the chain requests a no-response, the bracket terminates unconditionally when the last request of the chain that has EB in its first request processes.

If a chain contains both BB and EB, conditional bracket termination rule 1 states that the bracket unconditionally terminates when the last request of the chain processes, regardless of the form of response requested.

Unconditional bracket termination rule 2 states that a bracket terminates unconditionally when the last request of the chain that has EB in its first request processes, regardless of the form of response requested.

Additional bracket termination rules include the following:

- No more than one BB and one EB can be outstanding from a LU at one time.
- When CANCEL with EB is sent or received to terminate a chain that does not have EB, the CANCEL is treated as a definite response chain with EB and always ends the bracket regardless of the response type.
- The CHASE, LUSTAT, and QC data flow control requests may flow both in and between brackets. Each of these normal flow requests may have EB, but none may have BB.
- When brackets are used, only function management data requests with BB may flow between brackets.

Bracket Error Conditions

Three types of error conditions are associated with the management of brackets:

- Violations detected by the sender. Any attempt to transmit data in violation of bracket protocol is rejected (for example, if the first speaker attempts to send BB while a bracket is in process). The mechanism for passing this error information is a negative return code.
- Bracket protocol errors detected by the receiver due to sender errors or lack of LU bracket synchronization. The appropriate action is for the receiver to send the bracket state error sense code on a negative response to the other LU, and to discard the request.
- Errors detected by the receiver and caused by *race conditions*. A race condition occurs when both the host and the Series/1 try to begin a bracket at the same time. The appropriate action is for the receiver to send the bracket race error sense code on a negative response to the other LU and to discard the request. A retry of the operation may be necessary.

Segmenting

Series/1 SNA support accepts request/response units that the NCP segmented. Series/1 support does not segment outbound request/response units; Series/1 supports only inbound segmentation.

Half-Duplex Flip-Flop

Half-duplex flip-flop protocol restricts data transmission on the normal flow. It specifies that only one LU may send requests on the normal flow at a given time. The sending LU's session partner is prohibited from sending on the normal flow at the same time.

The protocol differs whether or not brackets are used:

Not using brackets

At session activation, one LU is designated the first sender, and the other LU is designated the first receiver. The sender issues normal flow requests, and the receiver issues responses. When the sender completes transmission, he can transfer control of sending by setting the change direction indicator on the last request sent.

Using brackets

At session activation, one LU is designated the first speaker, and the other LU is designated the bidder. When between brackets, each LU is in contention state and either can send; but in a race condition the contention winner is always the first speaker. When not between brackets, the LUs are subject to the protocol described when not using brackets.

Duplex

Duplex protocol allows both the primary and secondary session partners to send messages at the same time on the normal flow. All messages sent or received are single element chains. Each session partner always has the right-to-send.

Some restrictions apply when using duplex. SNA does not support message resynchronization. All messages pass as single element chains. SNA automatically sends responses to messages received to the host and does not allow the application to explicitly accept or reject messages.

Duplex is quite useful for batch transfers of large amounts of data when data can flow in both directions simultaneously. You can also use duplex for inquiry type sessions when you do not want to wait for a reply before requesting something else.

See "Message Transmission in Duplex Mode" on page 5-11 for a more detailed description of duplex protocols.

Pacing

Pacing allows a receiving connection point manager to control the data transfer rate on the normal flow. It prevents overloading a receiver with unprocessed requests when the sender generates requests faster than either the receiver or the network can process them.

Select pacing when establishing a session; it remains in effect for the duration of the session.

- If you select pacing for the Series/1 to network control program flow, all normal flow requests on that flow are paced.
- If you (or the host) select pacing for the network control program to Series/1 flow, all normal flow requests on that flow are paced.
- Expedited flow requests are never paced.

If you use pacing and session buffering, ensure that the values of the RECVBUF and SENDBUF parameters of the SNALU configuration statement correlate to the appropriate pacing value:

- Specify the pacing value for messages sent from the Series/1 LU to the NCP as a BIND command parameter. This BIND parameter is the secondary connection point manager (CPMGR) send pacing count. The pacing count specifies the number of messages that can be sent before receiving a pacing response. The Series/1 sends a pacing request for a pacing response from the NCP. Receipt of a pacing response indicates to the Series/1 that it can send another group of this LU's messages. A pacing group is the number of messages defined by the pacing count.
- The NCP also specifies the pacing count for messages sent to the Series/1 as a BIND parameter. This BIND parameter is the secondary connection point manager (CPMGR) receive pacing count. The NCP sends the number of messages specified by the pacing count. The NCP sends a pacing request for a pacing response from the Series/1. A pacing response indicates that the NCP can send another group of messages to the Series/1. Series/1 SNA support sends the pacing response when the number of buffers available for receiving messages is greater than or equal to the number of messages or message segments (when segmenting is required) to be received.

If the PU uses session buffering and if the number of messages or message segments received at one time for this LU is greater than RECVBUF, the Series/1 session terminates. This condition may cause suspension of the use of the primary LU. You can ensure that your session does not terminate as a result of receiving too many messages by choosing the value of RECVBUF so that it is greater than or equal to the receive pacing count. When segmenting is possible, the value of RECVBUF must be equal to or greater than the number of buffers required per message times the receive pacing count.

To illustrate, assume that the BIND secondary receive pacing count equals 7, and the number of segments per message is 3. Then, the number of free buffers that must be available before the NCP sends another pacing group is RECVBUF $\geq$ (7 x 3) = 21.

A pacing value of zero means no pacing is in effect and, therefore, no pacing response is sent. When you do not use pacing and the number of messages received

at one time for the LU is greater than RECVBUF and if the PU uses session buffering, the session terminates.

Only one pacing response is generated for each pacing request. In Series/1 SNA support, the Series/1 always responds to a pacing request with an isolated pacing response (IPR). Refer to your System/370 and NCP publications for information on how to tune your SNA network with respect to pacing and buffer requirements.

SNA Commands

This section contains the SNA commands that Series/1 SNA support or your program processes or sends, as indicated for each command. For convenience, the commands in this section appear in alphabetical order.

Figure 13-6 lists these commands by function. Following each session control and data flow control command is the corresponding request code for the command; following each function management data network services command is the network services header for the command.

Type	Command	Request Code
Session Control Commands	ACTLU	X'0D'
	ACTPU	X'11'
	BIND	X'31'
	CLEAR	X'A1'
	DACTLU	X'0E'
	DACTPU	X'12'
	SDT	X'A0'
	STSN	X'A2'
	UNBIND	X'32'
Data Flow Control Commands		
	BID	X'C8'
	CANCEL	X'83'
	CHASE	X'84'
	LUSTAT	X'04'
	QC	X'81'
	QEC	X'80'
	RELQ	X'82'
	RSHUTD	X'C2'
	RTR	X'05'
	SHUTC	X'C1'

Figure 13-6 (Part 1 of 2). Session

Type	Command	Request Code
	SHUTD	X'C0'
	SIG	X'C9'
Function Management Data Network Services Commands + (Configuration Services)		
	REQDISCONT	X'01021B'
Function Management Data Network Services Commands + (Session Services)		
	INIT-SELF	X'010681'
	NSPE	X'010604'
	TERM-SELF	X'010683'

Figure 13-6 (Part 2 of 2). Session

See Figure 13-1 on page 13-1 and Figure 13-2 on page 13-3 to determine if you should ever expect to receive any given command and if it is allowed for you to send a given command. If you are not allowed to send a command and you attempt to do so, SNA rejects the request and issues a negative return code.

ACTLU Command

The ACTLU command is an expedited flow command that activates an LU-SSCP session.

Series/1 SNA support processes your ACTLU command:

- If SNA finds no protocol violations and if the function management profile is 0, the transmission subsystem profile is 1, and:
  - You specify primary error recovery, then:
    - If you have an active LU-LU session at this time, the response is exception response (ERP); if not, the response is cold. The session remains active.
    - If you have an outstanding bind or unbind at this time, the response is cold, and SNA notifies you that the session terminated.
  - You do not specify error recovery, then:
    - If you have an active LU-LU session at this time, the response is cold, and SNA notifies you that the session terminated.
    - If you do not have an active LU-LU session at this time, the response is cold.
- If SNA finds protocol violations or if the function management profile is not zero or the transmission subsystem profile is not 1, SNA sends a negative response. SNA does not inform you of the receipt of the erroneous ACTLU.

ACTPU Command

The ACTPU command is an expedited flow command that activates a PU-SSCP session.

Series/1 SNA support processes your ACTPU command:

- If SNA finds no protocol violations and if the function management profile is 0, the transmission subsystem profile is 1, and:
  - You specify primary error recovery, then:
    - If the SSCP IDs are not equal and you specified them on the NETHOST SSCPID parameter, the response is a positive cold response. If you have an active LU-LU session, SNA notifies you that your session terminated.
    - If the SSCP IDs are not equal and you did not specify them on the NETHOST SSCPID parameter, the type of response sent depends on whether you have an active LU-LU session. If you have an active LU-LU session, the response is ERP; if not, the response is cold. The sessions remain active.
    - If the SSCP IDs are equal, the response is the same as if the SSCP IDs are not equal and you did not specify them on the NETHOST SSCPID parameter.
    - If SNA sends a request discontact in any of the above situations, SNA sends a negative response with an X'0801' sense code.
  - If you did not specify primary error recovery:
    - If the SSCP IDs are not equal whether or not you specified them on the NETHOST SSCPID parameter, the response is always negative with a X'0801' sense code.
    - If the SSCP IDs are equal and you have an active LU-LU session, the response is a positive cold response, and SNA notifies you that your session terminated.
    - If the SSCP IDs are equal and you do not have an active LU-LU session, the response is a positive cold response.
- If SNA finds protocol violations or if the function management profile is not 1, SNA sends a negative response. SNA does not inform you of the receipt of the erroneous ACTPU.

BID Command

The BID command is a normal flow command used by a primary LU to request permission from a secondary LU to begin a bracket.

SNA processes the BID command if you are in transmit mode. BID converts to a negative response with sense code X'0814' (bracket bid reject — RTR forthcoming), unless during session termination; in this case, the sense code is X'081C'.

Series/1 SNA support performs all appropriate data flow control processing.

BIND Command

The BIND command is an expedited flow command that specifies the session rules and protocols. The BIND command is further discussed in "Bind Processing" on page 15-2.

CANCEL Command

The CANCEL command is a normal flow command that terminates a partially sent chain.

You must process the CANCEL command:

- If you send CANCEL and violate any bracketing or chaining protocols, SNA rejects the request to send CANCEL with an appropriate return code.
- If you receive CANCEL, Series/1 SNA support performs the appropriate data flow control processing. If you receive CANCEL as an exception request, CANCEL passes to you with the sense code with which it appeared.

CHASE Command

The CHASE command is a normal flow command that ensures that there are no outstanding responses to requests on the normal flow.

You must process the CHASE command. You can send either a positive or negative response to CHASE.

Series/1 SNA support performs all appropriate data flow control processing.

CLEAR Command

The CLEAR command is an expedited flow command that resets all finite state machines associated with the LU-LU session.

Series/1 SNA support processes the CLEAR command. When you receive CLEAR, Series/1 SNA support immediately sends a positive response. After receiving and responding to CLEAR, SNA receives either UNBIND or SDT. In both situations (CLEAR/UNBIND or CLEAR/SDT) SNA sends you a negative return code. This code is from the NETPUT, NETGET, or NETCTL instruction and signifies that the session was either terminated or reset.

DACTLU Command

The DACTLU command is an expedited flow command that deactivates the LU-SSCP session and, if active, the LU-LU session.

Series/1 SNA support processes DACTLU for you:

- If it finds no protocol violations, SNA sends a positive response. If you have an active LU-LU session at this time, SNA notifies you that your session terminated.
- If it finds protocol violations, SNA sends a negative response. SNA does not inform you of the receipt of the erroneous DACTLU.

DACTPU Command

The DACTPU command is an expedited flow command that deactivates the PU-SSCP session and all LU-SSCP and LU-LU sessions that are active.

Series/1 SNA support processes DACTPU for you:

- If it finds no protocol violations, SNA sends a positive response. If you have an active LU-LU session at this time, SNA notifies you that your session terminated.
- If it finds protocol violations, SNA sends a negative response. SNA does not inform you of the receipt of the erroneous DACTPU.

INIT-SELF Command

The INIT-SELF command is a normal flow command that requests session initiation with a specific host application. Series/1 SNA constructs the INIT-SELF command and sends it to the host LU if NETINIT ACQUIRE=YES. If the INIT-SELF command is valid, the host sends a positive response.

LUSTAT Command

The LUSTAT command is a normal flow command that sends status information from one LU to its session partner.

You must process the LUSTAT command. Series/1 SNA support performs the appropriate data flow control processing when you send or receive (NETCTL) LUSTAT.

NSPE Command

The NSPE command is a normal flow command that notifies the Series/1 LU-SSCP session manager that an error occurred during a session initiation or session termination procedure.

Series/1 SNA support processes NSPE for you. When SNA receives NSPE during session initiation, it notifies you of the error by a negative return code from NETINIT. When SNA receives NSPE during session termination, it does not notify you of the error, and the session terminates.

QC Command

The QC command is a normal flow command that notifies the receiving LU that the sender has placed itself in a quiesce state.

You must process the QC command.

- If you send QC, you must have received the QEC command and have responded positively. Otherwise, SNA rejects the attempt to send QC with a negative return code.
- If you receive QC and have not sent the QEC command and received a positive response to it, SNA send a negative response with a sense code of X'0809' (mode inconsistency) to the primary LU. SNA does not notify you that the QC command was processed.

Series/1 SNA support performs all appropriate data flow control processing.

QEC Command

The QEC command is an expedited flow command that requests the receiving LU to stop sending and go into a quiesce state.

You must process the QEC command:

- If you send QEC, no special action is required.
- If you receive QEC, you must respond positively. You receive a negative return code if you attempt to send a negative response.

Series/1 SNA support performs all appropriate data flow control processing.

RELQ Command

The RELQ command is an expedited flow command that removes the receiving LU from the quiesce state.

You must process the RELQ command:

- If you send RELQ, Series/1 SNA support performs the appropriate data flow control processing. If you receive a positive response to RELQ, SNA activates the data flow control traffic. If you receive a negative response to RELQ, the data flow control traffic remains quiesced.
- If you receive RELQ, you must respond positively. You receive a negative return code if you attempt to send a negative response.

REQDISCONT Command

The REQDISCONT command is a normal flow command that notifies the SSCP to disconnect this Series/1 PU from the SNA network.

Series/1 SNA support processes REQDISCONT for you. SNA sends the REQDISCONT command after the operator enters a PUDACT or SNADACT command.

RSHUTD Command

The RSHUTD command is an expedited flow command that informs the primary LU that the secondary LU is at an end-of-job condition and that it would like the session terminated.

Series/1 SNA support performs the appropriate data flow control processing.

RTR Command

The RTR command is a normal flow command that permits the primary LU to begin a bracket.

You must process the RTR command. You can send RTR after you reject a data request containing a begin bracket or a BID command with sense code X'0814' (bracket bid reject – RTR forthcoming), by issuing NETGET or NETCTL TYPE=RTR while between brackets.

Series/1 SNA support performs all appropriate data flow control processing.

SDT Command

The SDT command is an expedited flow command that sets the data traffic finite state machine in the active state. Series/1 SNA support processes SDT for you.

SHUTC Command

The SHUTC command is an expedited flow command that indicates that the secondary LU completed session processing and placed itself in the quiesce state.

You must process the SHUTC command. Before you send SHUTC, you must have received a SHUTD command and responded positively.

Series/1 SNA support performs all appropriate data flow control processing.

SHUTD Command

The SHUTD command is an expedited flow command that requests that the secondary LU enter the quiesce state after it completes end-of-session processing.

You must process the SHUTD command. When you receive SHUTD, you must respond positively. You receive a negative return code if you attempt to send a negative response.

Series/1 SNA support performs all appropriate data flow control processing.

SIG Command

The SIG command is an expedited flow command that sends signal codes between logical units.

You must process the SIG command. You can send either a positive or negative response to SIG. You can send any signal code that is agreed to by you and the primary LU.

Series/1 SNA support performs all appropriate data flow control processing.

STSN Command

The STSN command is an expedited flow command sent by the primary LU conveying and/or requesting sequence number information from the secondary LU. Series/1 SNA support processes STSN for you and uses it during processing of the NETINIT instruction.

TERM-SELF Command

The TERM-SELF command is a normal flow command that requests the session to terminate.

UNBIND Command

The UNBIND command is an expedited flow command sent to terminate a session.

Chapter 14. SNA Command and Data Flows

This chapter shows examples of typical data and command exchanges that are possible during a session. An explanation of the flow exchange follows the examples.

The examples are organized in the following order:

- Establishing a session
- Sending messages to the host
- Receiving messages from the host
- Controlling message flow
- Terminating a session
- Processing an UNBIND HOLD command.

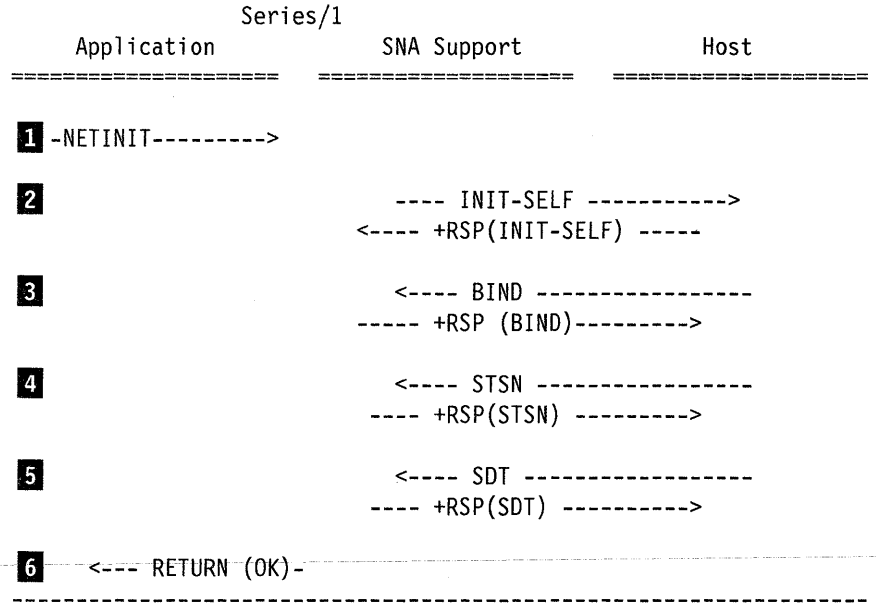
The following summarizes the abbreviations used:

BB	Begin bracket
BC	Begin chain
CD	Change direction
DR	Definite response
EB	End bracket
EC	End chain
ER	Exception response
FI	Format indicator
MC	Middle of chain
QC	Quiesce complete
QEC	Quiesce at end-of-chain
RELQ	Release quiesce
RSP	Response, either positive (+RSP) or negative (−RSP)
RTR	Ready to receive
RU	Request/response unit
SDT	Start data traffic.

Establishing a Session

SNA support processes the SNA commands in the following example to establish a session with the host.

Processing and Related Protocol



Example Explanations

1 The SNA application issues a NETINIT to establish a session with a host subsystem. If you specified the RESYNC=YES option, SNA processes the resynchronization data. If NETINIT fails, the operation completes immediately with an unsuccessful return code; the rest of the steps are not executed.

2 If requested by the application, SNA sends an INIT-SELF command to the host (ACQUIRE=YES); otherwise the support waits for a BIND.

If the command syntax is valid, the host returns a positive response. If the host sends a negative response, the operation completes immediately with an unsuccessful return code; the remaining steps are not executed.

3 The host sends a BIND command to start a session with the Series/1 LU.

The base system's SNA support performs basic BIND checking, and if successful, returns a positive response. If unsuccessful, the base system's SNA support sends a negative response, and the operation completes immediately with an unsuccessful return code; the rest of the steps are not executed.

4 If necessary for restart recovery, the host sends STSN with appropriate indicators and sequence numbers. The SNA support resynchronizes messages as described in Chapter 10, "Message Recovery and Resynchronization."

5 The host enables normal message flow by sending Start Data Traffic (SDT). The SNA support sends a positive response and completes the NETINIT operation functions.

6 The SNA support returns control to the application program.

Note: Any error at any step results in an appropriate response to the host and an error return to the application program.

Sending Messages to the Host

The examples in this section show sending (NETPUT) SNA command and data flow with and without message resynchronization.

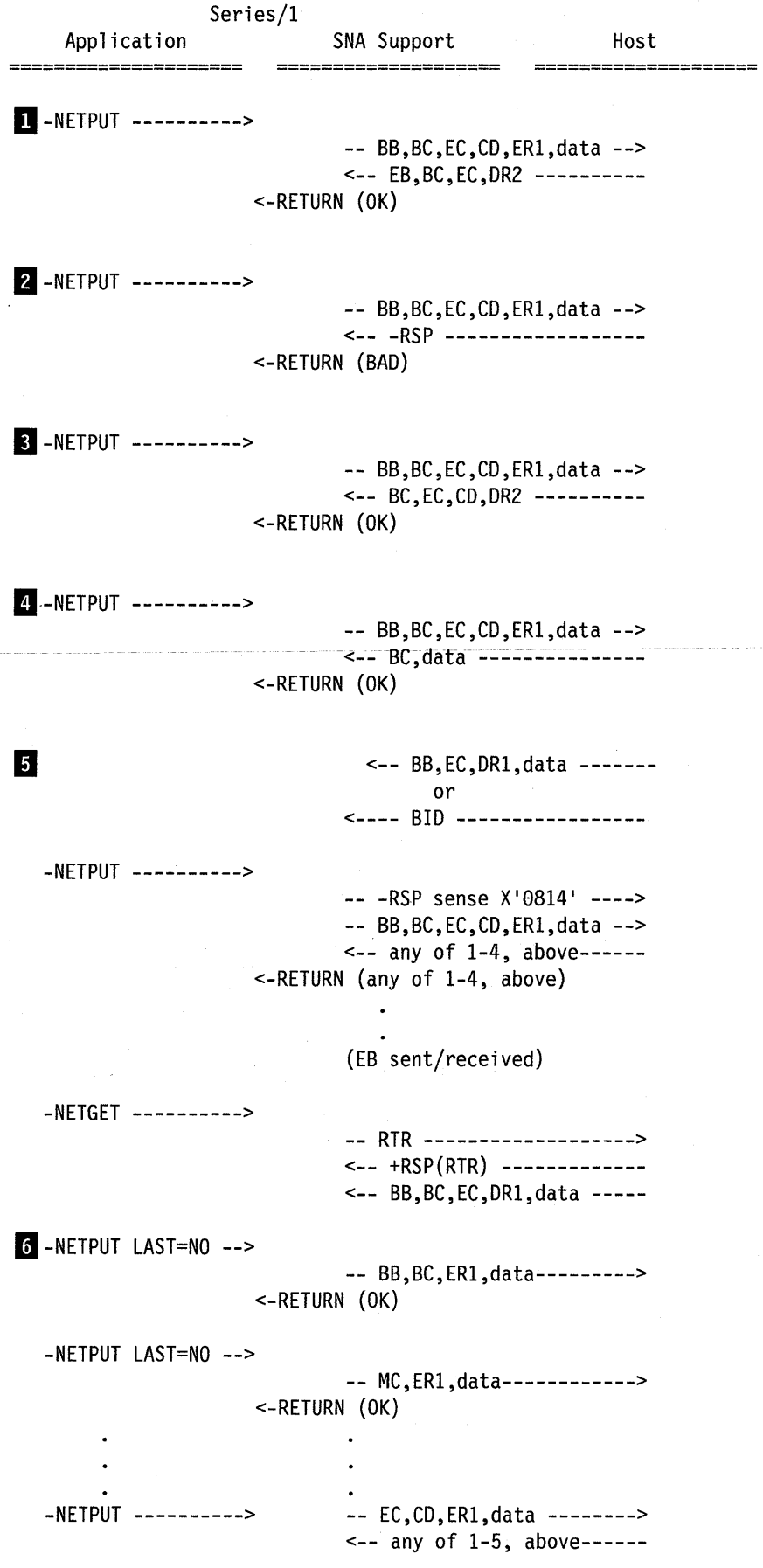
Sending Messages on Sessions with Resynchronization Support

The following examples show the processing and related protocol with various NETPUT operations executing in a resynchronous session.

These notes apply to all the examples:

- Unless otherwise shown, we assume that the SNA session is in the between-brackets state. Therefore, the Begin-Bracket (BB) indicator is shown on the transmit operations. If the SNA support were already in a bracket, the BB would not appear.
- SNA support always sets the Change-Direction (CD) indicator when transmitting in sessions with resynchronization support, unless you specify EOT=YES. The INVITE=YES option is therefore implied on each NETPUT, and an example showing INVITE=YES is not included.
- Including the FMH parameter causes SNA to set the Format-Indicator (FI); otherwise, the protocol is the same as shown.
- On any NETPUT, the length of the application's data can exceed the length of the RU. SNA blocks the data into multiple RUs as necessary (if allowed in session protocol), and handles all the necessary chaining.
- Any session-terminating errors received (for example, CLEAR/UNBIND) return to the application, as appropriate.

Processing and Protocol Data Flow with Resynchronization



```
                          <-RETURN (any of 1-5, above)

7 -NETPUT VERIFY=YES ->          -- BB,BC,EC,CD,DR1,data -->
                                <-- +RSP -----------------
                          <-RETURN (OK)

8 -NETPUT VERIFY=YES ->          -- BB,BC,EC,CD,DR1,data -->
                               <-- -RSP -----------------
                          <-RETURN (BAD)

9 -NETPUT EOT=YES------->        -- BB,EB,BC,EC,DR1,data -->
                               <-- (either of 7 or 8, above)
                          <-RETURN (either of 7 or 8, above)

10 -NETPUT EOT=YES,LAST=NO --> -- BB,EB,BC,ER1,data --->
        .                                 .
        .                                 .
    -NETPUT LAST=NO ---->    ------- MC,ER1,data ----->
        .                                 .
        .                                 .
    -NETPUT ------------>     -- EC,DR1,data --------->
                               <-- (either of 7 or 8, above)
                          <-RETURN (either of 7 or 8, above)


11 -NETGET----------->           <---- ¬EB,¬CD --------
                          <-RETURN (OK)

    -NETPUT ---------->
                          <-RETURN (BAD)
```

Example Explanations

1 The application issues a NETPUT. The host responds with an End Bracket (EB) indication, ending the transaction.

2 The application issues a NETPUT. The host responds with a negative response (−RSP) to the message. The application receives an unsuccessful completion return code.

3 The application issues a NETPUT. The host responds with a Change-Direction (CD) indication. (It expects to receive more data.)

4 The application issues a NETPUT. The host responds with an RU containing data. The application receives a successful completion return code.

5 The host requested to start a transaction prior to the NETPUT issued by the application. When the application issues a NETPUT, Series/1 SNA sends a -RSP with a sense code of X'0814', rejecting the host's request to start the transaction. Series/1 SNA then processes the NETPUT normally.

Upon completion of the NETPUT, the application receives a return code indicating the status of the NETPUT.

The application should issue a NETGET when between brackets, at which time SNA sends a Ready-To-Receive (RTR) command.

6 The application issues several NETPUT LAST=NO operations to include several data elements in the same message (chain). SNA blocks the data into a send buffer. If all the data fits into the buffer, control returns to the application (successful completion) without sending the buffer.

If all the data does not fit into a buffer, the filled buffer is transmitted as shown, and the rest of the data is moved into the next send buffer. The transmitted buffer contains the BB and BC indicators, as appropriate.

As subsequent buffers become filled, they are transmitted as middle-of-chain (MC) RUs. The final (LAST=YES) NETPUT sends the last buffer as end-chain (EC).

7 The application requests a response from the host. SNA requests a response by setting the DR1 indicator. The host responds with a positive response (+RSP). The application receives a successful completion return code.

8 The application requests a response from the host. SNA requests a response by setting the DR1 indicator. The host responds with a negative response (−RSP). The application receives an unsuccessful completion return code.

9 The application requests to end the transaction on this NETPUT. SNA ends the transaction by including the End-Bracket (EB) indicator. SNA also requests a definite response from the host to ensure the recoverability of the data.

Note: This option is valid only for IMS, although no validation is done by SNA.

10 The application requests to end the transaction after a series of NETPUT LAST=NO operations. Note that this option (EOT=YES) is valid only on the first NETPUT LAST=NO, since the EB indicator must be sent at the beginning of a chain. SNA ends the transaction by including the End-Bracket (EB) indicator. SNA also requests a definite response from the host at the end of the chain to ensure recoverability.

The remainder of SNA processing is the same as Item 6.

Note: This option is valid only for IMS, although no validation is done by SNA.

11 The application is receiving a message from the host and has not yet received the right-to-send or end-of-transaction indication. The application issues a NETPUT, which is rejected by SNA as an invalid operation.

Sending Messages on Sessions without Resynchronization

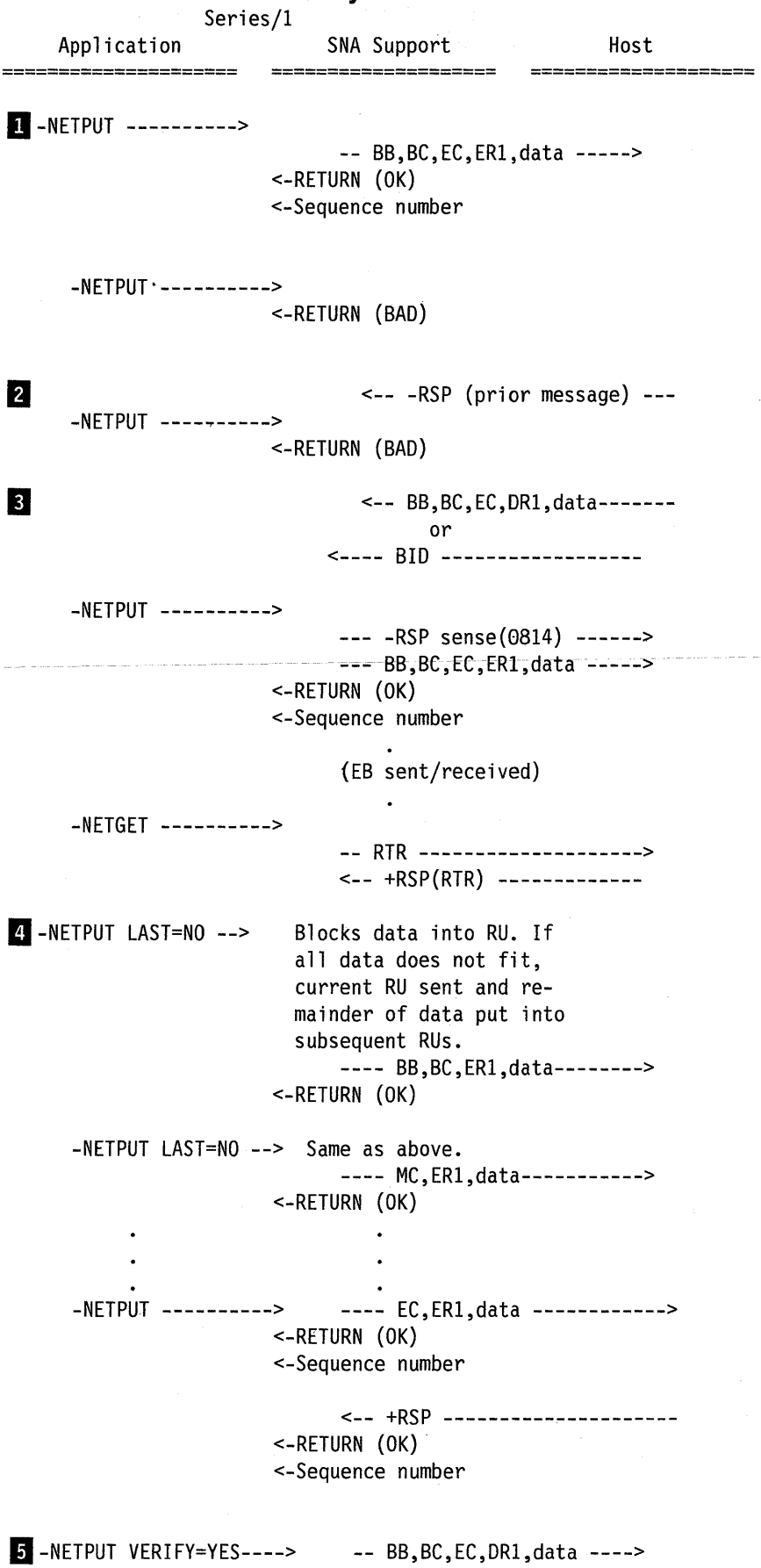
The following examples show the processing and related protocols with various NETPUT operations executing in a nonresynchronous session.

These notes apply to all the examples:

- Unless otherwise shown, the SNA session is assumed to be in the between-brackets state. Therefore, the Begin-Bracket (BB) indicator is shown on the transmit operations. If SNA were already in a bracket, the BB would not appear.
- Including the FMH parameter causes SNA to set Format-Indicator (FI); otherwise, the protocol is the same as shown.

- On any NETPUT, the length of the application data can exceed the length of the RU. SNA blocks the data into multiple RUs as necessary (if allowed in session protocol). SNA handles all the necessary chaining.
- The sequence number returned to the application is the sequence number of the first (or only) RU in the transmitted chain.
- Any session-terminating errors received (for example, CLEAR/UNBIND) return to the application, as appropriate.

Processing and Protocol Data Flow without Resynchronization



```
                                    <--- +RSP --------------------
                                <-RETURN (OK)
                                <-Sequence number

6 -NETPUT VERIFY=YES ->          -- BB,BC,EC,DR1,data ----->
                                <-  -RSP ------------------
                                <-RETURN (BAD)

 -NETCTL TYPE=RECV ---->
                                <-RETURN (-RSP RECEIVED)

7  -NETPUT EOT=YES---->      --BB,EB,BC,EC,ER1,data --->
                                <-RETURN (OK)
                                <-Sequence number

8 -NETPUT EOT=YES,LAST=NO -->   -- BB,EB,BC,ER1,data --->
         .                                .
         .                                .
     -NETPUT LAST=NO ---->     -- MC,ER1,data ----------->
         .                                .
         .                                .
     -NETPUT ------------>     -- EC,ER1,data ----------->
                                <-RETURN (OK)
                                <-Sequence number


9 -NETGET----------->            <- ¬EB,¬CD --
                                <-RETURN (OK)

   -NETPUT ---------->
                                <-RETURN (BAD)

10 -NETPUT INVITE=YES---->      -- BB,BC,EC,CD,ER1,data -->
                                <-RETURN (OK)
                                <-Sequence number
```

Example Explanations

1 The application issues a NETPUT. SNA sends the data to the host as an exception response chain (one or more RUs). Return goes to the application before it receives any reply or response from the host.

The application receives the sequence number of the first (or only) RU in the chain sent to the host.

2 The host sent a −RSP to some data previously sent from the application. The application issues a NETPUT. The application receives a bad return code. No data is processed for the NETPUT.

The application must issue NETCTL TYPE=RECV to receive the sense code and the failing sequence number.

3 The host requested to start a transaction prior to the NETPUT operation issued by the application. When the application issues a NETPUT operation, SNA sends a -RSP with a sense code of X'0814', denying permission for the host to begin the transaction. SNA then processes the NETPUT operation normally.

Upon completion of the NETPUT operation, the application receives a return code indicating the status of the NETPUT. The application receives the sequence number of the message sent to the host.

The application should issue a NETGET when next between brackets, at which time SNA sends a Ready-To-Receive (RTR) command.

4 The application issues several NETPUT LAST=NO operations to include several data elements in the same message (chain). SNA blocks the data into a send buffer. If all the data from the NETPUT LAST=NO fits into the buffer, control returns to the application (indicating successful completion) without sending the buffer.

If all the data does not fit into a buffer, the filled buffer is transmitted as shown, and the remainder of the data is moved into the next send buffer. The transmitted buffer contains the BB and BC indicators as appropriate.

As subsequent buffers become filled, they are transmitted as middle-of-chain (MC) RUs. The final (LAST=YES) NETPUT sends the last buffer as end-chain (EC).

The application receives the sequence number of the first (or only) RU in the chain sent to the host.

5 The application requests a response from the host. SNA requests a response by setting the DR1 indicator. The host responds with a positive response (+RSP).

The application receives a return code indicating successful completion.

The application receives the sequence number of the first (or only) RU in the chain sent to the host.

6 The application requests a response from the host. SNA requests a response by setting the DR1 indicator. The host responds with a negative response (−RSP).

The application receives a return code indicating unsuccessful completion. The application must issue a NETCTL TYPE=RECV to receive the failing sense code, and sequence number.

7 The application requests to end the transaction on this NETPUT. SNA ends the transaction by including the End-Bracket (EB) indicator. SNA sends the data to the host as an exception response chain (one or more RUs). Return is made to the application before any reply or response is received from the host.

The application receives the sequence number of the first (or only) RU in the chain sent to the host.

8 The application requests to end the transaction after a series of NETPUT LAST=NO operations. Note that this option (EOT=YES) is valid only on the first NETPUT LAST=NO, since the EB indicator must be sent at the beginning of a chain. SNA ends the transaction by including the End-Bracket (EB) indicator.

The rest of SNA processing is the same as Item 6, above.

9 The application is receiving a message from the host and has not yet received the right-to-send or end-of-transaction indication. The application issues a NETPUT operation. SNA rejects the NETPUT operation as an invalid operation.

10 The application desires that the host be granted permission to send. SNA grants permission by setting the CD indicator. SNA sends the data to the host as an exception response chain (one or more RUs). Return is made to the application before any reply or response is received from the host.

The application receives the sequence number of the first (or only) RU in the chain sent to the host.

Receiving Messages from the Host

The following examples show processing and related protocol involved with various NETGET operations.

These notes apply to all the examples:

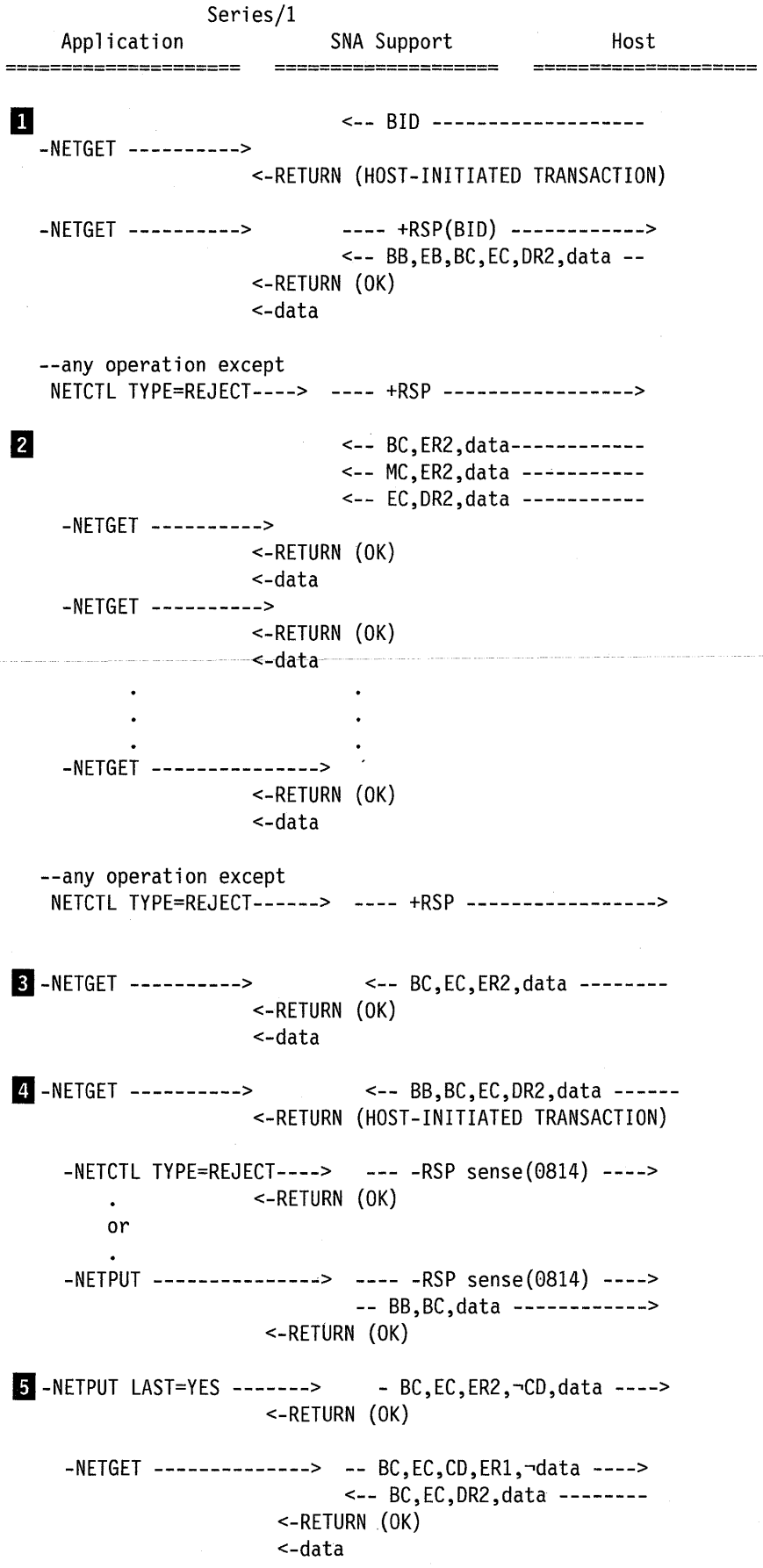
- All requests from the host ask for a DR2 or ER2 response from the Series/1. The processing is the same if the host requests DR1, or both DR1 and DR2. SNA support sends a response on behalf of the application, specifying the type of response requested by the host on its last request.

  SNA sends a positive response on the next operation issued if a response is necessary. SNA sends a negative response only if you issue a NETCTL TYPE=REJECT.

  SNA never sends a positive response to an exception response request.

- SNA handles all data deblocking from RUs into application buffers. Applications need not be involved with the correlation of data lengths to RU sizes; SNA manages this.

Processing and Related Protocol



```
    --any operation except
     NETCTL TYPE=REJECT---->   ---- +RSP ---------------->

6 -NETGET ---------->          <-- BC,EC,CD,DR2,¬data ---
                      <-RETURN (OK)

    --any operation except
     NETCTL TYPE=REJECT---->   ---- +RSP ---------------->

7 -NETGET ---------->          <-- EB,BC,EC,DR2,¬data ---
                      <-RETURN (OK)

    --any operation except
     NETCTL TYPE=REJECT---->  --- +RSP ---------------->

8 -NETGET ---------->          <-- BC,EC,DR2,¬data-------
                      <-RETURN (OK)

    --any operation except
  -NETCTL TYPE=REJECT---->   ---- +RSP ---------------->

9
                              <-- -RSP(RTR) -------------
    -NETGET ---------->
                      <-RETURN (HOST-INITIATED TRANSACTION)
```

Example Explanations

1 The application issues a NETGET, and a BID command arrives from the host. The BID can arrive before or after the NETGET.

The NETGET fails with a "host initiated transaction" return code. The application then issues a NETGET, sending a positive response to the BID command. (A NETPUT would result in a negative response with sense code X'0814' being sent to the BID command.)

In addition to the return code, the data returns to the user in the designated area. In this example, the NETGET is long enough to receive all the data in the message.

SNA sends a positive response (+RSP) to the host on the next operation from the application unless you issue NETCTL TYPE=REJECT. SNA sends +RSP only if the host requests a definite response (DR2). If exception response (ER2) is indicated, the +RSP is not sent.

2 The host sends a long message, requiring more than one RU and more than one NETGET operation. The arrival of the RUs can be before the first NETGET or interspersed with the NETGET operations. If SNA needs more data to satisfy a particular NETGET operation, it waits for the next RU's arrival and processing before the NETGET completes.

Note that the last NETGET completes with the end of message.

SNA sends a positive response (+RSP) to the host on the next operation from the application unless you issue NETCTL TYPE=REJECT. SNA sends +RSP only if the host requests a definite response (DR2). If exception response (ER2) is indicated, the +RSP is not sent.

3 The host sends a single element chain with no bracket indicators. The message can arrive before or after the NETGET operation. The operation completes with the end of message indication. The data moves into the designated area.

SNA sends a negative response (-RSP) to the host if you issue NETCTL TYPE=REJECT. Since an exception response (ER2) is indicated, a +RSP is not sent.

4 The host starts a transaction with data. The NETGET operation completes with "host initiated transaction."

The application decides to reject the host transaction. This can be done with NETCTL TYPE=REJECT or a NETPUT to start a transaction. SNA sends a negative response, containing the sense code X'0814', to the host.

5 The application issues a NETGET while sending data (in brackets). This occurs if the NETGET follows a NETPUT operation in a non-resynchronous session. SNA sends a null RU to the host, specifying CD. The host responds with a standalone message, as in Item 3.

6 The application issues a NETGET, and the host sends a null RU with CD. The RU can arrive before or after the NETGET. No data passes to the application.

SNA sends a positive response to the host as in Item 2.

7 The application issues a NETGET operation, and the host sends a null RU with EB. The RU can arrive before or after the NETGET. No data passes to the application.

SNA sends a positive response to the host as in Item 2.

8 The application issues a NETGET, and the host sends a null RU without bracket or CD. The RU can arrive before or after the NETGET. No data passes to the application.

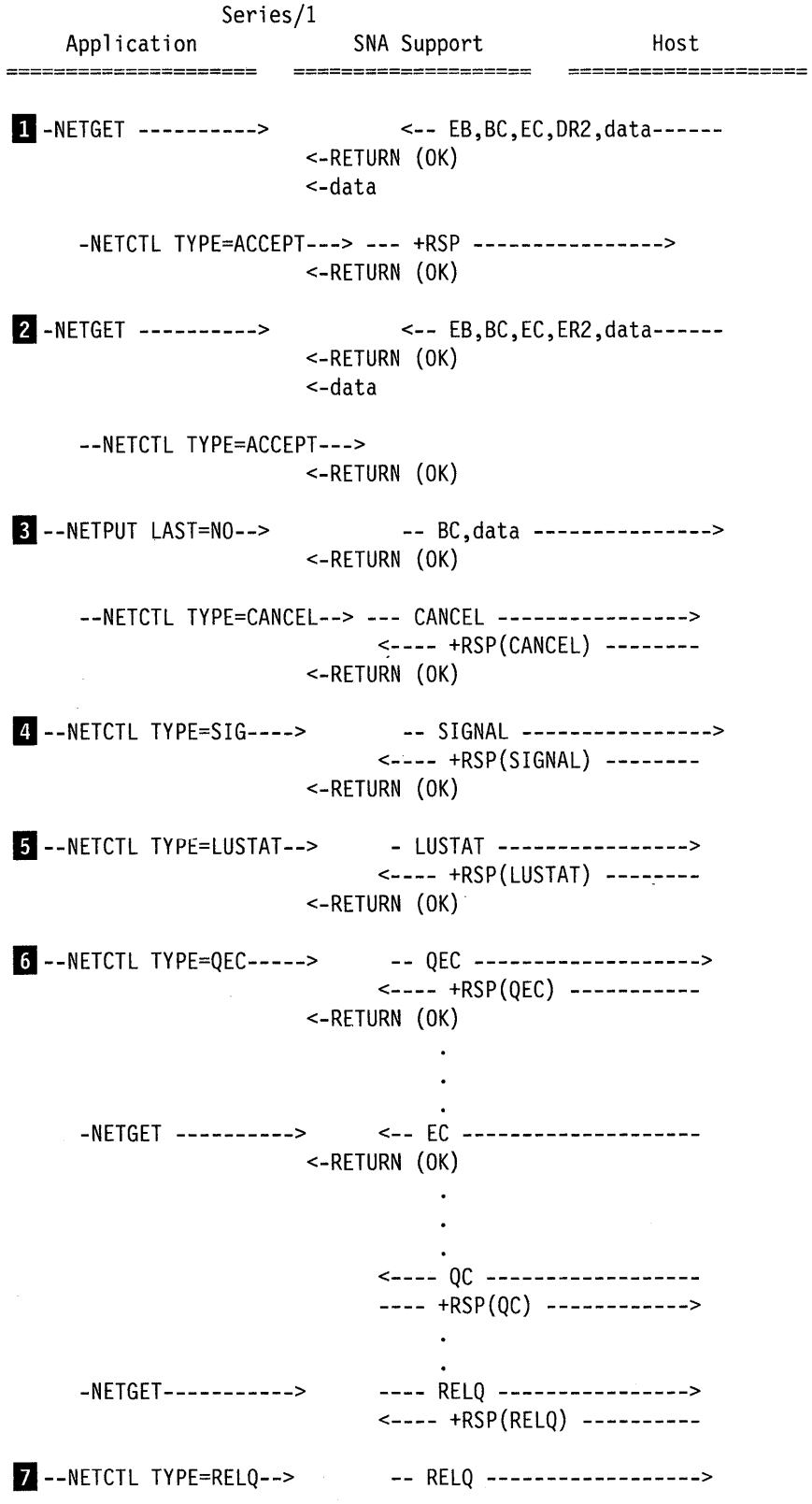
SNA sends a positive response to the host as in Item 2.

9 The host sends a negative response to an RTR previously sent. (The host cancels the transaction it tried to start earlier, which was then rejected by SNA.) No data passes to the application. The return code is "no message available" (−25).

Controlling Message Flow

The following examples show processing and related protocol involved with various NETCTL operations.

Processing and Related Protocol


Series/1
Application
SNA Support
Host
1 -NETGET ---------->
<-- EB,BC,EC,DR2,data------
<-RETURN (OK)
<-data
-NETCTL TYPE=ACCEPT---> --- +RSP ---------->
<-RETURN (OK)
2 -NETGET ---------->
<-- EB,BC,EC,ER2,data------
<-RETURN (OK)
<-data
--NETCTL TYPE=ACCEPT--->
<-RETURN (OK)
3 --NETPUT LAST=NO-->
-- BC,data ---------->
<-RETURN (OK)
--NETCTL TYPE=CANCEL--> --- CANCEL ---------->
<---- +RSP(CANCEL) --------
<-RETURN (OK)
4 --NETCTL TYPE=SIG---->
-- SIGNAL ---------->
<---- +RSP(SIGNAL) --------
<-RETURN (OK)
5 --NETCTL TYPE=LUSTAT-->
- LUSTAT ---------->
<---- +RSP(LUSTAT) --------
<-RETURN (OK)
6 --NETCTL TYPE=QEC----->
-- QEC ---------->
<---- +RSP(QEC) -----------
<-RETURN (OK)
-NETGET ---------->
<-- EC ----------
<-RETURN (OK)
<---- QC ----------
---- +RSP(QC) ---------->
-NETGET---------->
---- RELQ ---------->
<---- +RSP(RELQ) ----------
7 --NETCTL TYPE=RELQ-->
-- RELQ ---------->">

```
                                <---- +RSP(RELQ) ----------
                            <-RETURN (OK)

8 --NETCTL TYPE=RTR--->           -- RTR ------------------->
                                <---- +RSP(RTR) -----------
                            <-RETURN (OK)

9 <-- LUSTAT ----------------
  --any operation---->          ---- +RSP ---------------->
                            <-RETURN (BAD)
  --NETCTL TYPE=RECV-->
                            <-RETURN (OK)
```

Example Explanations

1 The application finishes receiving a message and acknowledges its receipt by issuing NETCTL TYPE=ACCEPT. SNA transmits a positive response to the host.

2 The application finishes receiving a message sent by the host as an exception response chain and acknowledges its receipt by issuing NETCTL TYPE=ACCEPT. SNA does not send a positive response.

3 The application transmits data with NETPUT LAST=NO, and decides to terminate the operation before the normal end of data. The application issues a NETCTL TYPE=CANCEL.

SNA sends a CANCEL command to the host if at least one RU was transmitted for the NETPUT LAST=NO. Control returns to the application when it receives the CANCEL response.

4 The application issues a NETCTL TYPE=SIG, asking the host to grant the right to send. If the application does not have the right to send, SNA sends a SIGNAL to the host with a signal code of X'0001'. Control returns to the application when it receives the SIGNAL response.

5 The application sends status information to the host with NETCTL TYPE=LUSTAT. The status information is contained in a specified user area. SNA sends the status information in a LUSTAT command. Control returns to the application when it receives the LUSTAT response.

6 The application issues NETCTL TYPE=QEC, asking the host to stop sending data at the end of the current message. SNA sends a Quiesce-at-End-of-Chain (QEC) command to the host. Control returns to the application when it receives the response.

The application then continues to receive the current message. The host sends Quiesce-Complete (QC) at the end of the chain; SNA responds to the QC with a +RSP.

When the application is ready to receive the next message from the host, it issues a NETGET. SNA sends a Release Quiesce (RELQ) to the host and handles the response. SNA then processes the NETGET as it receives data from the host.

7 The application issues NETCTL TYPE=RELQ, informing the host that it can again send messages to the SNA application. This operation follows a NETCTL TYPE=QEC, once the application is again ready to receive messages. SNA sends a Release-Quiesce (RELQ) command to the host. Control returns upon receipt of the response.

8 The application issues NETCTL TYPE=RTR, informing the host that it is ready to receive the next message. SNA sends a Ready-To-Receive (RTR) command to the host. Control returns upon receipt of the response.

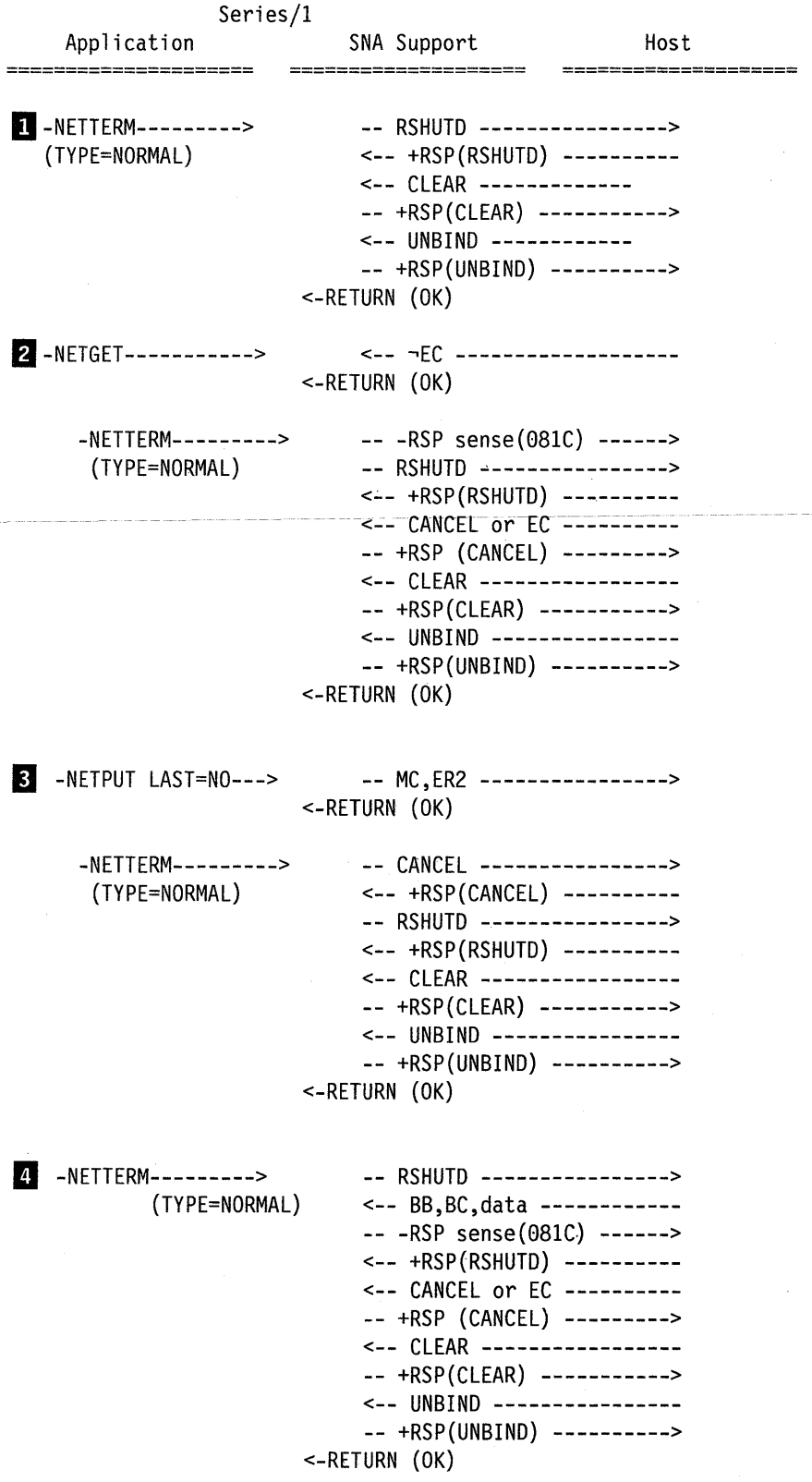
9 The application was notified on some prior operation that status is available to receive. The application issues NETCTL TYPE=RECV to get the status.

The return code of the NETCTL TYPE=RECV specifies the type of status. See "Receiving Status" on page 8-5 for the types of status. Additional status information (for example, sense code on a negative response received) returns in the specified area.

Terminating a Session

The following example shows processing and related protocol involved with various NETTERM operations.

Processing and Related Protocol



```
5                          <-- SHUTD -----------------
                                -- +RSP(SHUTD) ----------->

     -NETPUT----------->
                          <-RETURN (BAD)

     -NETCTL TYPE=RECV->
                          <-RETURN (OK)
          .                               .
          .                               .
          .                               .
     -NETTERM--------->        -- SHUTC ----------------->
      (TYPE=NORMAL)            <-- +RSP(SHUTC) ----------
                               -- RSHUTD ----------------->
                               <-- +RSP(RSHUTD) ----------
                               <-- CLEAR ----------------
                               -- +RSP(CLEAR) ----------->
                               <-- UNBIND ----------------
                               -- +RSP(UNBIND) ---------->
                          <-RETURN (OK)

6  -NETPUT------------>        -- BC,EC,ER1,data -------->
                          <-RETURN (OK)

     -NETTERM----------->      -- CHASE ----------------->
      (TYPE=NORMAL)            <-- +RSP(CHASE) ----------
                               -- RSHUTD ----------------->
                               <-- +RSP(RSHUTD) ----------
                               <-- CLEAR ----------------
                               -- +RSP(CLEAR) ----------->
                               <-- UNBIND ----------------
                               -- +RSP(UNBIND) ---------->
                          <-RETURN (OK)

7  -NETTERM----------->
            (TYPE=QUICK)       -- TERM-SELF ------------->
                               <-- +RSP ------------------
                               <-- UNBIND ----------------
                               --- +RSP(UNBIND) --------->
                          <-RETURN (OK)

8  -NETTERM----------->
            (TYPE=IMMED)       -- UNBIND ---------------->
                          <-RETURN (OK)
```

Example Explanations

1 The application issues a NETTERM operation to terminate its session with the host. SNA handles all the indicated protocol on behalf of the application.

The NETTERM completes after the receipt of the UNBIND.

Note: The host may or may not send the CLEAR command.

2 The application issues NETTERM while in the process of receiving data, but has not yet completed with an end-of-message return code. SNA considers this an abnormal termination situation. SNA performs error processing by sending a negative response to the host with a sense code of X'081C' (Request not Executable).

Then SNA transmits RSHUTD to ask for session termination.

The NETTERM completes after receipt of the UNBIND. The return code indicates that SNA sent the negative response to the host during termination processing.

Note: The host may or may not sent the CLEAR command.

3 The application issues NETTERM while in the process of transmitting a message to the host. The normal end of message was not issued by the application. SNA considers this an error situation. SNA performs error processing by sending a CANCEL command to the host to abnormally terminate the current chain.

If SNA has not yet started transmitting the chain, the current buffer is discarded, and the CANCEL is not sent. This situation happens if the application issues a small number of short-length NETPUT LAST=NO commands, and not enough data has been NETPUT to fill an RU.

Then SNA transmits RSHUTD to ask for session termination.

The NETTERM completes after receipt of the UNBIND. The return code indicates that SNA sent the CANCEL command to the host during termination processing.

Note: The host may or may not send CLEAR command.

4 Once the application issues a NETTERM operation, SNA responds negatively to any message received from the host. The sense code used is X'081C' (Request not Executable). The return code notifies the application of any negative response.

Then SNA transmits RSHUTD to ask for session termination.

The NETTERM completes after receipt of the UNBIND.

Note: The host may or may not send the CLEAR command.

5 The host sends SHUTD to initiate orderly termination. SNA responds to the SHUTD and notifies the application.

The application must issue NETCTL TYPE=RECV to determine the type of status received.

The application completes its processing, then issues a NETTERM. SNA sends a SHUTC command to the host.

SNA transmmits RSHUTD to ask for session termination.

The NETTERM completes after receipt of the UNBIND.

Note: The host may or may not send the CLEAR command.

6 The application is executing on a nonrecoverable session. The last operation issued by the application prior to the NETTERM is a NETPUT without the VERIFY parameter, causing SNA to send the last RU as an exception request.

To prevent the last message (or messages) from being lost, SNA sends a CHASE command to the host. Then, SNA waits for a response to the CHASE before continuing.

Then SNA transmits RSHUTD to ask for session termination.

The NETTERM completes after receipt of the UNBIND.

Note: The host may or may not send the CLEAR command.

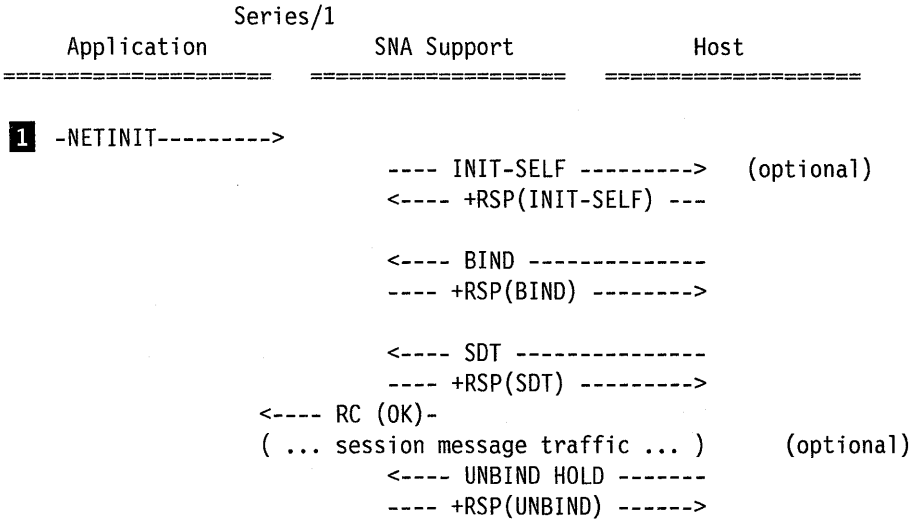
7 The Series/1 SNA application issues a NETTERM TYPE=QUICK operation to unconditionally terminate the session with the host. The SNA support sends a TERM-SELF command to the host SSCP. Once UNBIND is received from the host, control is returned to the application, after freeing session resources

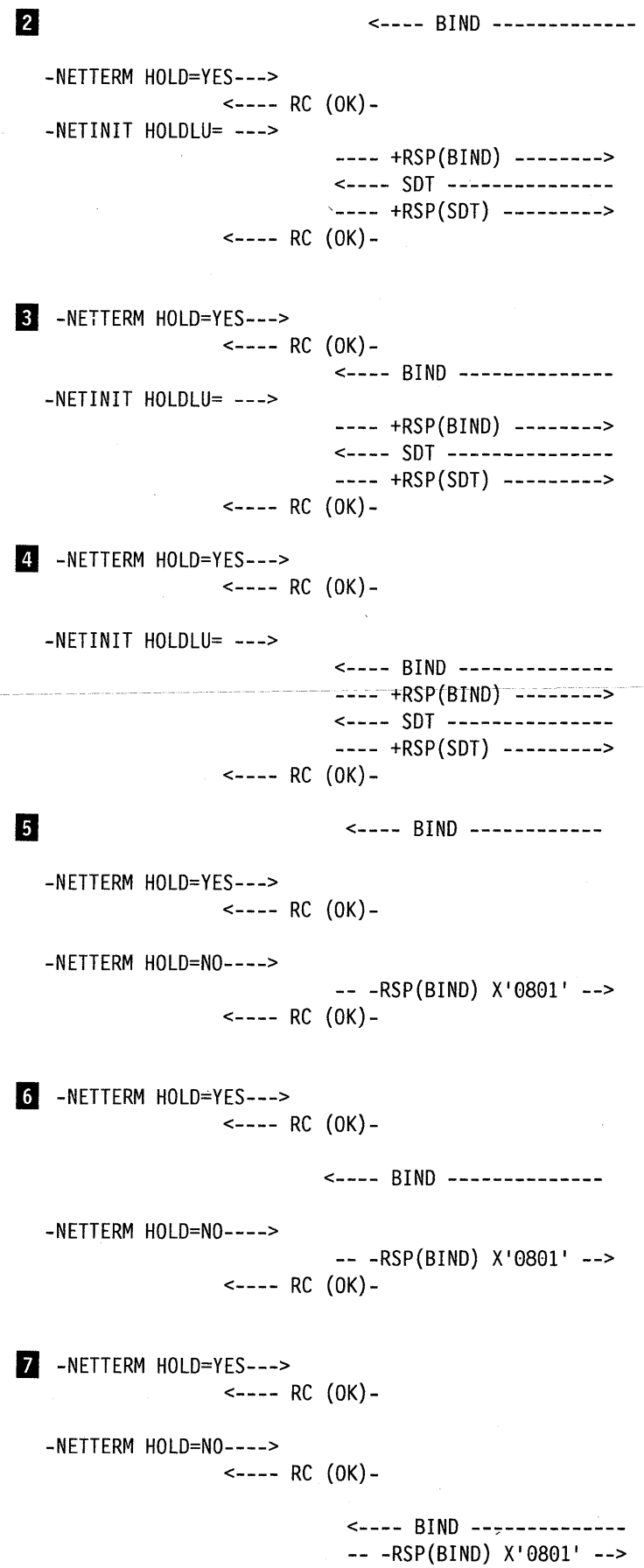
8 The Series/1 SNA application issues a NETTERM TYPE=IMMED operation to unconditionally terminate the session with the host. The SNA support sends an UNBIND command to the host and returns control to the application, after freeing session resources. The response to the UNBIND is accepted if sent from the host, but it is not required.

Processing the UNBIND HOLD

The example marked 1 shows a session being established and receiving an UNBIND HOLD. The examples marked 2 through 4 show the processing and related protocol for establishing a session after receiving this UNBIND HOLD. The processing and protocol in the example marked 5 through 9 shows the session being terminated after receiving the UNBIND HOLD.

Processing and Related Protocol




2
<---- BIND ----------
-NETTERM HOLD=YES--->
<---- RC (OK)-
-NETINIT HOLDLU= --->
---- +RSP(BIND) -------->
<---- SDT ---------------
---- +RSP(SDT) --------->
<---- RC (OK)-
3
-NETTERM HOLD=YES--->
<---- RC (OK)-
<---- BIND --------------
-NETINIT HOLDLU= --->
---- +RSP(BIND) -------->
<---- SDT ---------------
---- +RSP(SDT) --------->
<---- RC (OK)-
4
-NETTERM HOLD=YES--->
<---- RC (OK)-
-NETINIT HOLDLU= --->
<---- BIND --------------
---- +RSP(BIND) -------->
<---- SDT ---------------
---- +RSP(SDT) --------->
<---- RC (OK)-
5
<---- BIND ------------
-NETTERM HOLD=YES--->
<---- RC (OK)-
-NETTERM HOLD=NO---->
-- -RSP(BIND) X'0801' -->
<---- RC (OK)-
6
-NETTERM HOLD=YES--->
<---- RC (OK)-
<---- BIND --------------
-NETTERM HOLD=NO---->
-- -RSP(BIND) X'0801' -->
<---- RC (OK)-
7
-NETTERM HOLD=YES--->
<---- RC (OK)-
-NETTERM HOLD=NO---->
<---- RC (OK)-
<---- BIND ------------
-- -RSP(BIND) X'0801' -->

```
8                                  <---- BIND --------------

  -NETTERM HOLD=NO---->
                                  -- -RSP(BIND) X'0801' -->
                     <---- RC (OK)-


9  -NETTERM HOLD=NO---->
                     <---- RC (OK)-

                                  <---- BIND --------------
                                  -- -RSP(BIND) X'0801' -->
```

Example Explanations

1 The SNA application issues a NETINIT to establish a session with a host subsystem. After the session is established, the host sends an UNBIND HOLD. The SNA application sends a positive response to the host.

2 The Series/1 SNA support receives a BIND before the SNA application issues a NETTERM HOLD=YES. When the SNA application issues NETTERM HOLD=YES, Series/1 SNA support returns a successful return code. The SNA application then issues a NETINIT HOLDLU=, which sends a positive response and establishes the session.

3 The SNA application issues a NETTERM HOLD=YES before receiving a BIND. After the SNA application issues a NETINIT HOLDLU=, it sends a positive response to the BIND and establishes the session.

4 The SNA application issues a NETTERM HOLD=YES and NETINIT HOLDLU= before receiving the BIND. When the application receives the BIND, it establishes the session and completes the NETINIT.

5 Series/1 SNA support sends a BIND before the SNA application issues a NETTERM HOLD=. The NETTERM HOLD=YES completes successfully, after which the application issues NETTERM HOLD=NO. This causes SNA to send a negative response to the BIND. A successful return code returns to the SNA application.

6 The SNA application issues a NETTERM HOLD=YES. When the application receives the BIND, it issues a NETTERM HOLD=NO. This causes SNA to send a negative response.

7 Prior to Series/1 SNA support receiving a BIND, the SNA application issues a NETTERM HOLD=YES followed by NETTERM HOLD=NO. Both NETTERM operations complete successfully. When the application receives the BIND, Series/1 SNA support sends a negative response to the BIND.

8 The SNA application issues a NETTERM HOLD=NO causing SNA to send a negative response to a previously received BIND.

9 The SNA application issues NETTERM HOLD=NO prior to Series/1 SNA support receiving a BIND. When SNA receives the BIND, it sends a negative response.

Passthru Examples

Example 1

SNA issues both the primary and secondary sides of the passthru session before establishing the actual session, setting up the passthru link between the remote LUs. SNA issues the primary NETOPEN first and then issues the secondary NETINIT instruction, using the ACB from the primary NETOPEN to complete the connection.

```
         NETOPEN   ACB=PRIACB,                                    C
                   •
                   •
                   •
                   PTHRU=YES

         NETINIT   LU=LUNUM,                                      C
                   HOSTID=DEST,                                   C
                   RESYNC=NO,                                     C
                   ATTNEV=AEVENT,                                 C
                   PACB=PRIACB,                                   C
                   PTHRU=YES
         •
         •
         •
PRIACB   DATA  A(*-*)
LUNUM    DATA  F'1'
DEST     NETHOST ISAPPID=CICS,ISMODE=INQUIRY
AEVENT   ECB   0
```

Example 2

SNA issues both the primary and secondary sides of the passthru session before establishing the actual session, setting up the passthru link between the remote LUs. SNA issues the secondary NETINIT first and then issues the primary NETOPEN instruction, using the LUNUM from the secondary NETINIT to complete the connection.

```
         NETINIT   LU=LUNUM,                                    C
                   HOSTID=DEST,                                 C
                   RESYNC=NO,                                   C
                   ATTNEV=AEVENT,                               C
                   PTHRU=YES

         NETOPEN   ACB=PRIACB,                                  C
                   •
                   •
                   •
                   PACB=LUNUM,                                  C
                   PTHRU=YES

         •
         •
         •
PRIACB   DATA  A(*-*)
LUNUM    DATA  F'1'
DEST     NETHOST ISAPPID=CICS,ISMODE=INQUIRY
AEVENT   ECB   0
```

Example 3

In this example, SNA opens the secondary side of the passthru session first using NETINIT, linking the remote primary LU to the passthru link. Because the remote primary LU always begins the session, (by sending a BIND command without receiving an INIT-SELF), SNA does not establish the primary side of the passthru session until the remote primary LU sends the BIND. The NETINIT is issued before the BIND is received. The attention event is posted when SNA sends the BIND across the passthru link. SNA holds the BIND until it issues the primary NETOPEN or it issues NETTERM for the session.

```
         NETINIT   LU=LUNUM,                                    C
                   HOSTID=DEST,                                 C
                   RESYNC=NO,                                   C
                   ATTNEV=AEVENT,                               C
                   PTHRU=YES

         WAIT      AEVENT
         MOVE      POSTCODE,AEVENT
         RESET     AEVENT
         IF        (POSTCODE,EQ,-1)
           NETOPEN   ACB=PRIACB,                                C
                   •
                   •
                   •
                   PACB=LUNUM,                                  C
                   PTHRU=YES

         ELSE
           NETTERM LU=LUNUM
         ENDIF
         •
         •
         •
PRIACB   DATA  A(*-*)
LUNUM    DATA  F'1'
DEST     NETHOST ISAPPID=CICS,ISMODE=INQUIRY
AEVENT   ECB   0
```

Chapter 15. Series/1 SNA Instruction Processing

This chapter describes the base Series/1 SNA commands and subsequent protocol processing generated when the SNA application program instructions are executed. The description also includes a discussion of the verifications made to ensure that you adhere to the base system protocols when you specify various parameter options and combinations of these options.

The Series/1 SNA application program instructions discussed are:

NETINIT	Establishing a session
NETPUT	Sending messages to the host
NETGET	Receiving messages from the host
NETCTL	Controlling message exchange
NETTERM	Terminating a session.

NETINIT - Establishing a Session

Series/1 SNA support handles the establishing of an LU to LU session for the application program when it issues the NETINIT instruction.

Session Initiation

An application can initiate a session with the host when issuing a session establishment operation (NETINIT ACQUIRE=YES). Series/1 SNA support sends the SNA Initiate-Self (INIT-SELF) command to the host in this case. Otherwise, the session establishment operation causes Series/1 SNA to wait for the host to initiate an LU-LU session by sending an SNA Bind (BIND) command.

You can use NETINIT also to establish a passthru session.

The Series/1 SNA application must specify the host ID data list (NETHOST) for the session when issuing the NETINIT.

When the Series/1 SNA application issues a NETINIT, it can provide an event name to be posted when an attention event occurs. When the attention event is posted, the application should issue a receive (NETGET) operation, unless you specified passthru support. If you specified passthru, you must also issue NETOPEN for the primary side of the passthru session. See "Opening a Passthru Session" on page 15-4.

For Nonpassthru Sessions

Bind Processing

The host always sends the SNA Bind (BIND) command to start a session. Series/1 SNA support accepts or rejects the BIND according to the "basic BIND checking" of the base system SNA support. If SNA rejects the BIND, the host passes a return code to the application program indicating the session establishment operation failed. After SNA successfully processes this command, the LUs are said to be "in session" and are referred to as "session partners."

The Series/1 SNA application can request that the BIND be passed to the Series/1 application as data (NETINIT SESSPRM=). The BIND data is available only when the basic bind checking of the base support is successful. The BIND command passes as received, excluding the RU code. The actual byte length of the BIND replaces the RU code. For more information on BIND and BIND checking refer to Chapter 13, "SNA Protocols."

Certain BIND parameters affect the SNA formats and protocols used by Series/1 SNA and the host when sending and receiving messages and commands. The following is a summary of the BIND parameters referenced by Series/1 SNA during the session.

- The base system SNA support requires some BIND parameters (basic BIND checking). Series/1 SNA accepts any BIND acceptable to the base system SNA support by sending a positive response to the host.
- The base system Series/1 SNA support accepts the following BIND parameters, but SNA does not support the setting of the corresponding RH indicators on messages to the host; they are not reported on messages from the host.
  - Alternate code
  - End-user cryptography.
- In addition to the SNA protocols enforced by Series/1 SNA base support, special considerations for SNA usage of the following BIND parameters apply:

  FM and TS Profiles

  In the discussion of SNA functions in this book, we assume that you use FM Profile 4 and TS Profile 4. When the session is not bound with these parameters, the following considerations apply:

  FM Profile 3 An attempt to send the SNA command Quiesce at End of Chain (QEC) (NETCTL TYPE=QEC) or Release Quiesce (RELQ) (NETCTL TYPE=RELQ) results in an error return code.

  TS Profile 3 The host does not support message resynchronization. All sessions begin as if cold-started on both the primary-to-secondary and the secondary-to-primary flows.

Brackets

In the discussion of SNA functions in this book, we assume that you use brackets and that the secondary LU (SLU) may send EB. When the session is not bound with these parameters, the following considerations apply:

- SLU cannot send EB — A request to send a message with EB (NETPUT EOT=YES) results in an error return code.
- No brackets are used — The BB and EB indicators are never set by SNA. A request to send a message with EB (NETPUT EOT=YES) results in an error return code.

Response Protocols

In the discussion of SNA functions in this book, we assume that the SLU may send either RQE or RQD chains. When the session is not bound with this parameter, the following considerations apply:

- SLU can send only RQE chains — all messages are sent RQE. A request to send a message with verification (NETPUT VERIFY=YES) results in an error return code.
- SLU can send only RQD chains — all messages are sent RQD. The performance of the application may be impacted by this condition.
- SLU can send only RQN chains — all messages are sent RQN. A request to send a message with verification (NETPUT VERIFY=YES) results in an error return code.

Chains

In the discussion of Series/1 SNA functions in this book, we assume that the SLU can send multiple element chains. When the session is not bound with this parameter, the following consideration applies:

- SLU can send only single element chains — a request to send a message of length greater than the RU-size determined from the BIND results in an error return code.

Session Restart

When establishing a session, Series/1 SNA handles the message resynchronization processes used to resynchronize message flow for restarted sessions that request resynchronization support (NETINIT RESYNC=YES or INIT).

Note: The Series/1 application is responsible for opening and closing the resynchronization data sets.

The host can send the SNA Set and Test Sequence Numbers (STSN) command as part of session restart. The STSN command establishes the message flow status of restarted sessions. SNA support handles STSN for the application. See Chapter 10, "Message Recovery and Resynchronization" for information on STSN processing.

Enabling Normal Message Flow

The host enables the session for normal flow data and commands by sending the SNA Start Data Traffic (SDT) command. Series/1 SNA support handles SDT for the application. When Series/1 SNA support processes the SDT command, the session establishment operation (NETINIT) completes.

For Passthru Sessions

Opening a Passthru Session

Passthru establishes a logical connection between a remote LU within a PU type 4 or 5 device attached to the Series/1 by SNA to a remote LU within a PU type 2 device attached to the Series/1 by Primary SNA. During the passthru session the Series/1 is transparent to both remote LUs. To establish a session, you must issue NETOPEN for the primary side and NETINIT (with PTHRU=YES) for the secondary side of the passthru session. NETINIT for the secondary side establishes the logical link with the PU type 4 or 5 remote LU, while NETOPEN for the primary side establishes the logical link with the PU type 2 remote LU.

You establish the link between the primary and secondary sides of the passthru session using the passthru access control block (PACB) parameter. You must omit this parameter from the first NETOPEN/NETINIT and you must specify it on the second NETOPEN/NETINIT for the passthru session. The passthru ACB is either the ACB returned if the first request was NETOPEN or the PU/LU number if the first request was NETINIT. Note that it does not matter whether the primary or secondary side of the passthru session opens first, as long as you use the PACB parameter on the second NETOPEN/NETINIT.

You can establish the passthru link immediately by issuing both NETOPEN and NETINIT to establish the connection. But if the Series/1 passthru application wishes to make the most use of its session resources and it knows which remote LU is going to start the actual session, the application can issue a NETOPEN/NETINIT for one side of the passthru link, wait on the BIND event/Attention event for the remote LU to try and start the session, and then issue the second NETOPEN/NETINIT. The passthru link is then established, and the actual session between the two remote LUs can continue.

For example:

The remote primary LU starts the session by sending the BIND without waiting for the remote secondary LU to send an INIT-SELF.

- The Series/1 passthru application issues NETINIT PTHRU=YES to establish communications with the remote LU.
- The application waits on the attention event specified on the NETINIT instruction.
- The remote primary LU sends the BIND command, at which time the attention event is posted with "BIND received."
- The passthru application issues NETOPEN PTHRU=YES,PACB=the NETINIT PU/LU #, establishing the passthru link.
- The BIND command passes to the remote secondary LU, which can either accept it or reject it.

If the remote secondary LU starts the session first, the application issues NETOPEN PTHRU=YES, waits on the BIND event to be posted when the remote secondary LU sends INIT-SELF, and issues NETINIT PTHRU=YES,PACB=NETOPEN ACB.

Most NETINIT parameters are either invalid or ignored when you specify passthru. You must specify RESYNC=NO, LUSWAIT must be YES, and you must not specify MSGDATA and SESSPRM. SNA ignores ACQUIRE and FULLDPX and you should not specify them. You must specify HOSTID, but SNA does not use it since it ignores ACQUIRE. You can specify ERRCODE and MSGPRIO, and SNA will use them for passthru sessions. The ATTNEV event is not required but is highly recommended. SNA posts the event it specifies with unique post codes reflecting when the passthru link establishes a session, when the session ends, or when an error occurs within the passthru link. If not used, the passthru application cannot obtain any status about the passthru link or the session it is supporting.

For the secondary side of the passthru link, you may issue only NETINIT and NETTERM. NETPUT, NETGET, and NETCTL are not allowed and SNA will reject them.

NETPUT - Sending Messages to the Host

Before the application can transmit any messages to the host, you must establish a session and enable the normal flow. After the NETINIT operation completes successfully, you can send messages to the host. The application programmer issues a send (NETPUT) operation to send messages to the host subsystem.

Sending Chains

The length of data associated with a single send operation is not limited by the RU size parameter specified in the BIND command. To send a message of length greater than this BIND parameter, SNA support creates a chain of separate (RUs). SNA sends a message of a length less than or equal to the BIND RU-size parameter as a chain of a single RU. The host subsystem treats a chain of RUs as a single message when received.

The application can control when the chain of RUs ends; that is, it can issue more than one send operation per chain. SNA does not send the end of chain (EC) indicator with the message unless you specify NETPUT LAST=YES.

SNA holds, as necessary, data specified on send operations for which EC is not to be sent (NETPUT LAST=NO) and does not transmit until a data length equal to the base system SNA support RU size is reached.

You can cancel a message being sent at any time before you send EC by issuing a control operation (NETCTL TYPE=CANCEL). This operation results in SNA sending an SNA Cancel (CANCEL) command. When the host receives CANCEL, it discards any part of the message it has already received.

Sending Function Management Headers

The data to be sent can contain one or more SNA headers at the beginning of the message. These headers are called function management headers (FMHs) and are created and controlled by the host subsystem and the SNA application. When sending messages with headers, the application supplies the header in the data and indicates its presence on the send operation (NETPUT FMH=YES). The FMH=YES parameter is valid only at the beginning of a message (the first NETPUT). Series/1 SNA does not perform any checks on the headers.

Sending and Brackets

Just as data RUs are grouped into larger units called "chains," chains can be grouped into larger units called "brackets." The nature of the chains within the bracket depends on which host subsystem is the session partner of the SNA application.

A transaction processing program can process a bracket in a host subsystem. The method used to select a specific transaction processing program on a host subsystem depends on which subsystem is the session partner of an SNA application. However, in the discussion of transactions in this book, we assume that the initiation of a host transaction processing program by the SNA application is associated with the beginning of a bracket being sent to the host.

The host subsystems define specific types of transactions that have meaning only in the context of communication with that subsystem. The method used by the transaction processing program to process a bracket depends on the type of transaction associated with the program. See Appendix D, "Host Subsystem Considerations" for more information.

When no bracket is in progress on the session, either LU may attempt to start a bracket. Series/1 SNA support sends Begin Bracket (BB) on a message when it issues a send (NETPUT) operation and no bracket is in progress. The Series/1 SNA application is always defined as "contention winner" on the session BIND command. Therefore, an SNA application BB "wins" over a simultaneous host subsystem attempt to start a bracket by an SNA Bid (BID) command or by sending a message with BB. A return code to indicate this condition passes to the Series/1 SNA application on the NETPUT that rejects the host bracket attempt.

Either the host or the Series/1 SNA application can end a bracket. When the Series/1 SNA application specifies the end of a transaction on a send (NETPUT EOT=YES) operation, the Series/1 SNA support sends End Bracket (EB) on the message to the host. When the host subsystem ends the bracket, it indicates it in the return code to a receive operation (NETGET) or a control operation (NETCTL TYPE=RECV).

When the bracket ends and Series/1 SNA support previously rejected a BID or a BB message from the host, the Series/1 SNA application can issue a receive (NETGET) operation to allow the host subsystem to retry the rejected bracket.

Sending with Half-Duplex Flip-Flop

Within a bracket, the Series/1 SNA support uses half-duplex flip-flop (HDX-FF) protocols to control which LU can send and which must receive.

To send messages and normal flow commands, the LU must be in send state; that is, have the right-to-send. The HDX-FF protocols determine the send/receive state of the LU. The LU that started the bracket initially has the right-to-send.

The protocol requires that the LU take turns being in send state. The sender grants the right-to-send to its partner by sending the change direction (CD) indicator with the message. On sessions for which resynchronization support has been requested (NETINIT RESYNC=YES or INIT), SNA sends all messages with CD or EB.

On sessions that do not request resynchronization support (RESYNC = NO), the application sends a CD by performing a receive (NETGET) operation or by an option on a send operation (NETPUT INVITE = YES). Note that sending a CD by performing a receive may result in SNA sending an extra message (null RU) carrying CD to the host.

If, while the Series/1 SNA application program is sending, Series/1 SNA support receives an SNA Signal (SIG) command from the host requesting CD, a return code indicating "status available" (−17) passes to the application. The application should then issue a control (NETCTL TYPE = RECV) operation to determine the status condition. When the control (NETCTL TYPE = RECV) operation indicates that a request for the right-to-send (SIG) was received, the application may honor the request by sending the CD.

Responses to Messages Sent

Series/1 SNA support can use both definite and exception response protocols when sending messages to the host subsystem. When using definite response protocol, the host must respond, either positively or negatively, to the message sent. When using exception response protocol, the host responds only to error conditions and sends only negative responses.

Series/1 SNA support uses exception response protocol except when the application specifies verification of the message on the send (NETPUT VERIFY = YES) operation. In this case, SNA requests a definite response. The operation does not complete until Series/1 SNA receives a positive or negative response from the host.

A positive response indicates the successful receipt of an acceptable message. A negative response indicates the successful receipt of an unacceptable message. The host subsystem sends a negative response when it finds an error in the message from the SNA application program or when an unusual condition in the host prevents the subsystem from accepting the message. Negative responses from the host to messages sent by the Series/1 SNA application carry information (sense data) describing the nature of the error condition.

When a return code indicates "status available" (−17), the Series/1 SNA application should issue a control (NETCTL TYPE = RECV) operation. If the return code indicates that a negative response was received, the sense data carried by the negative response returns to the Series/1 SNA application. The Series/1 SNA application should then issue a receive (NETGET) operation to receive an error message from the host subsystem. For more on responses, see "NETCTL - Controlling Message Exchange" on page 15-10.

NETGET - Receiving Messages from the Host

Before the application can receive any messages from the host, it must establish a session and enable the normal flow. The Series/1 SNA application issues a receive (NETGET) operation to obtain messages from the host subsystem.

Receiving Chains

The length of a message received by a single receive operation is not limited to the RU size parameter specified in the session BIND command. The host can use SNA chain protocol to send longer messages.

Chained RUs from the host subsystem are considered to be a single message. The data passes to the Series/1 SNA application until the end of the chain is reached or the application input area is filled. The actual data length also returns to the application.

When the chain length is greater than the length of the input buffer provided on the receive operation, Series/1 SNA holds the remaining data and passes it to the application on subsequent receive operations. When all of the data in the message has been passed, SNA sets a return code to notify the Series/1 SNA application that the end-of-message has been reached.

If the host subsystem sends an SNA Cancel (CANCEL) command to the Series/1 SNA support, the receive operation terminates with a "status available" (−17) return code. The application should then issue a control (NETCTL TYPE=RECV) operation to determine the status condition. When the "message cancelled" (33 or 34) condition is indicated, the application should discard any portion of the message already received.

Receiving Function Management Headers

Data received from the host may contain one or more SNA function management headers at the beginning of the message. A return code indicates the presence of these headers to the Series/1 SNA application. The headers pass to the application along with the received data when SNA issues a receive operation. Series/1 SNA support does not check the headers.

Receiving and Brackets

When the host attempts to start a bracket by sending a message with a Begin Bracket (BB) or an SNA BID command, a receive operation fails with a return code indicating that the host initiated a transaction. Since the Series/1 is always defined as the "contention winner" on the session BIND command, the Series/1 SNA application can choose whether to allow the transaction to begin.

Issuing a receive (NETGET) operation in this case allows the transaction to begin. SNA sends a positive response to the host to indicate that the transaction can begin. If the host initiates the transaction with a message with BB, the message data passes to the Series/1 SNA application. If the host initiates the transaction with an SNA BID command, the NETGET operation does not complete until the Series/1 SNA application receives the first message of the transaction. When it arrives, the message data passes to the Series/1 SNA application.

Receiving with Half-Duplex Flip-Flop

If the Series/1 SNA application is receiving and must send a message to the host, issuing a control (NETCTL TYPE=SIG) operation causes Series/1 SNA support to send the SNA Signal (SIG) command to the host. A return code passes to the application on a Series/1 SNA receive (NETGET) or control (NETCTL TYPE=RECV) operation when CD is received.

Responding to Messages Received

Messages from the host can arrive as definite response requested (RQD) or as exception response requested (RQE). Series/1 SNA support sends a positive response to RQD messages received from the host subsystem unless the Series/1 SNA application specifies a negative response.

A return code indicates the end of an RQD message being passed to the application. Then, any valid Series/1 SNA operation on the session (other than a message reject, NETCTL TYPE=REJECT) results in a positive response from SNA to the host. The Series/1 SNA application can issue a control operation (NETCTL TYPE=ACCEPT) to explicitly cause SNA to send the positive response to the host.

SNA can send a negative response (NETCTL TYPE=REJECT) after receiving any portion of the message. After receiving the end-of-message, SNA must send the negative response before issuing any other operation. After sending a negative response, the Series/1 SNA application should issue a receive operation (NETGET) to determine the action taken by the host subsystem.

When the application receives an RQE message, the considerations above for RQD message responses apply, except that SNA sends no response when a positive response is indicated.

Attention Event

You can specify an attention event on the session establishment (NETINIT ATTNEV=) operation to determine when a message arrives from the host. Series/1 SNA support posts the attention event for every Begin Chain request arriving from the host. When SNA posts an attention event, you should issue a receive (NETGET) operation to determine the nature of the host request.

Specifying an attention event affects the way SNA handles a receive operation. When between transactions, receive operations may complete with a "no messages available" (−25) return code whether or not you specified an attention event. However, within a transaction, a receive operation waits for a message to arrive if you did not specify an attention event for the session.

It is recommended that you use an attention event for sessions with host-initiated transactions or for sessions in which the Series/1 SNA application may be quiesced.

Figure 15-1 summarizes the handling of receive operations by Series/1 SNA with respect to attention events.

	Between transactions	In transaction
Attention event not specified	The NETGET does not wait for the message to arrive.	The NETGET waits for the message to arrive.
Attention event specified	The NETGET does not wait for the message to arrive.	The NETGET does not wait for the message to arrive.

Figure 15-1. NETGET and Attention Events

NETCTL - Controlling Message Exchange

When the host sends status information or an error indication, Series/1 SNA support passes a return code. The application should then issue a control (NETCTL TYPE=RECV) operation to determine the nature of the status condition.

If the application program needs to send status or an error indication to the host, use the appropriate control operation.

Negative Responses to Messages

The SNA application can reject the current message from the host subsystem by issuing a control (NETCTL TYPE=REJECT) operation after receiving any portion of a message. SNA support sends a negative response to the host.

The Series/1 SNA application can indicate that sense data is to be included with the negative response. Specify the sense data in a data area provided by the application when issuing the control operation. If the application does not provide sense data, Series/1 SNA sends a negative response with a "request not executable" sense code (X'081C') or, for rejecting a host initiated transaction, "bid reject— RTR forthcoming" (X'0814').

The host sends a negative response when it detects an error or some unusual condition in a message from the Series/1 SNA application program. When Series/1 SNA support receives a negative response, it sends the SNA Cancel (CANCEL) command, if necessary, to the host. Sense data accompanies a negative response to describe the type of error. The sense data returns to the Series/1 SNA application when it issues the control (NETCTL TYPE=RECV) operation.

For sessions established without resynchronization support (NETINIT RESYNC=NO), the number assigned to each message transmitted by a Series/1 SNA send (NETPUT) operation optionally returns to the application. Since, for non-resynchronization sessions, SNA passes the message number of the rejected message along with the status data, the application can determine the message in error.

Session Termination Request

In some cases, the host subsystem may request that the session be terminated. The Series/1 SNA support indicates this condition to the application with a return code and handles the termination protocol. The application should send any remaining messages to the host and then issue a termination (NETTERM) operation for Series/1 SNA to release the session resources. (See "NETTERM - Terminating a Session" on page 15-12 for more information on this subject.)

Status Information

When the host subsystem has status information, it sends an SNA Logical Unit Status (LUSTAT) command to Series/1 SNA support. Series/1 SNA support passes a status available (−17) return code to the Series/1 SNA application. The application should then issue a control (NETCTL TYPE=RECV) operation. When the return code indicates status information received, the sense data received from the host passes to the application.

When the Series/1 SNA application needs to send status information, you should issue a control (NETCTL TYPE=LUSTAT) operation. Series/1 SNA support sends the sense data from the status area with an SNA LUSTAT command.

Suspending Transmission

When one LU does not want or cannot receive any more data from the other LU, quiesce protocol suspends transmission in one direction.

When receiving, the Series/1 SNA application performs this protocol by issuing a control (NETCTL TYPE=QEC) operation to request the host to withhold sending messages. Series/1 SNA support sends the SNA Quiesce-at-End-of-Chain (QEC) command to the host subsystem.

When the Series/1 SNA application is ready to receive more messages, issue a receive (NETGET) operation or a control (NETCTL TYPE=RELQ) operation. Series/1 SNA support sends the SNA Release Quiesce (RELQ) command to the host to allow it to send again. While quiesced, the host does not send any messages to the Series/1 SNA application.

Conversely, the host subsystem can place the Series/1 SNA application into a quiesced state by using the same SNA command protocol. Any further requests from the application program to send messages or normal flow commands after being quiesced results in an error return code. A "status available" (−17) return code passes to the application when it receives RELQ from the host. The return code from the control (NETCTL TYPE=RECV) operation issued to receive status indicates that the host has released the quiesced state.

Requesting the Right-to-Send

When the Series/1 SNA application does not have the right-to-send, it can issue a control (NETCTL TYPE=SIG) operation to request the host to grant it. Series/1 SNA sends an SNA Signal (SIG) command to the host. The Series/1 SNA application must continue to receive until a message arrives with Change Direction (CD) or End Bracket (EB).

In like manner, the host can request the right-to-send. When Series/1 SNA receives a SIG command from the host, a "status available" (−17) return code passes to the Series/1 application. When you issue the control (NETCTL TYPE=RECV) operation, the return code indicates the request for the right-to-send.

Cancelling Messages

When a message starts but the end of message (NETPUT LAST=YES) has not yet been issued, the Series/1 SNA application can cancel the message with a control (NETCTL TYPE=CANCEL) operation. Series/1 SNA sends the SNA CANCEL command. The Series/1 SNA application retains the right-to-send.

Similarly, the host can cancel a partially sent message. The control (NETCTL TYPE=RECV) operation return code indicates that the message from the host was cancelled.

Ready-to-Receive

After sending a message indicating the end of a transaction (NETPUT EOT=YES) to the host or receiving an indication that the host ended a transaction, the Series/1 SNA application can indicate that it is ready to receive a transaction from the host by issuing a control (NETCTL TYPE=RTR) operation. The application should then issue a receive (NETGET) operation.

NETTERM - Terminating a Session

Session termination releases an LU from its current logical connection. The LU is then available for session with another LU. Either the host or the Series/1 SNA application can request termination at any time. SNA handles the termination protocol. System resources held to support the session are released during session termination.

The host can request either "orderly" or "unconditional" session termination. The return code notifies the Series/1 SNA application of the type of session termination request. The SNA application can initiate orderly or unconditional session termination procedures.

No messages can be exchanged on a session for which a successful termination operation has been issued. Messages received from the host after the termination operation is issued are rejected by the Series/1 SNA support. The negative responses sent by SNA to these messages carry a "request not executable" sense code (X'081C').

Orderly Session Termination

Orderly termination procedure allows normal processing to complete before the session terminates. The host requests orderly termination by sending the SNA Shutdown (SHUTD) command. SNA passes a return code to the application program when this occurs.

When notified by return code that orderly session termination was requested by the host, the application should send any remaining messages to the host and then issue a termination operation (NETTERM TYPE=NORMAL). SNA support sends the SNA Shutdown Complete (SHUTC) command.

The Series/1 SNA application can also call for orderly termination by issuing a session termination operation (NETTERM TYPE=NORMAL) before the host sends SHUTD. In this case, Series/1 SNA support sends the SNA Request Shutdown (RSHUTD) command to prompt the host to terminate the session. (Note that RSHUTD does not request the host to send SHUTD. The host sends UNBIND.)

Unconditional Session Termination

The host can unconditionally terminate the session by sending the SNA Unbind (UNBIND) command. Immediate session termination is indicated by return code to the current Series/1 SNA operation. The application should then issue an SNA session termination operation (NETTERM TYPE=NORMAL) to release system resources held to support the session. No messages can be sent to the host when the host has requested immediate session termination.

The Series/1 application program can call for unconditional session termination by issuing a NETTERM with either TYPE=QUICK or TYPE=IMMED. If you specify TYPE=QUICK, the Series/1 SNA support sends a TERM-SELF SNA command to the host. This command goes to the SSCP component of VTAM; VTAM then terminates the session with the host LU by sending an UNBIND to the Series/1.

If you specify TYPE=IMMED, the Series/1 SNA support sends an UNBIND command to the host and frees the Series/1 resources immediately. No further messages or SNA commands are accepted from the host. Some host systems may not process the receiving of an UNBIND from a secondary LU. For example, MVS VTAM must be at ACF Release 1.3 or later for this option to operate correctly. Consult your host system programmer to determine if TYPE=IMMED can be used.

Summary of Commands/Indicators Used

Figure 15-2 is a summary of SNA command and indicator processing by the Event Driven Executive SNA program product.

SNA Command/ RH Indicator	Sent to Host	Received from Host
INIT-SELF	X	
BIND		X
STSN		X
SDT		X
BB/EB	X	X
BC/EC	X	X
CD	X	X
FI	X	X
DR1I	X	X
DR2I		X
ERI/RTI/SDI	X	X
LUSTAT	X	X
SIGNAL	X	X
QEC	X	X
QC	X	X
RELQ	X	X
CANCEL	X	X
CHASE	X	X
BID		X
RTR	X	
RSHUTD	X	
SHUTD		X
SHUTC	X	
TERM-SELF	X	
UNBIND	X	X

Figure 15-2. SNA Commands/Indicators Used

Chapter 16. Overview of SNA Remote Job Entry

The $RJESNA utility is a licensed program that enables the Series/1, together with an EDX operating system using Systems Network Architecture (SNA), to communicate with a host system.

Within the SNA environment, the Series/1 operates as an SNA remote job entry workstation. You can create job streams on the Series/1 with the EDX editing facilities and transmit these job streams to the host for processing. Upon completion of the job stream processing, the host can either transmit the output to the Series/1 workstation or direct it to an output device on the host. The $RJESNA utility enables you to transfer data to and from the host without writing an SNA application program to perform these functions. Further, the processing capability of a large host system can be made available to you at the Series/1.

The $RJESNA utility provides easy-to-use operator commands that allow you to transmit and receive data, and control the operation of the utility. The commands are discussed in "$RJESNA Operator Commands" on page 16-8.

The $RJESNA utility attempts to ensure the integrity of all data sent to and received from the host. For example, a positive acknowledgment of the receipt of a data set is not sent until the last record has been correctly written to the Series/1 data set. Similarly, if, while transmitting a job stream, $RJESNA receives a negative response from the host, $RJESNA "aborts" the entire data stream. This procedure ensures that a partially submitted job stream is not scheduled for execution on the host.

The $RJESNA utility also supports the EDX spool function. In order for you to use this feature of $RJESNA, the spool function must be active in the Series/1. Refer to the *IBM Series/1 Event Driven Executive Commercial Applications Development Guide* or the *Operator Commands and Utilities Reference* for information on the use of the EDX spool function.

$RJESNA Hardware and Software Requirements

The minimum Series/1 hardware configuration for $RJESNA consists of:

- Series/1 processor (128K bytes minimum)
- SDLC Communications Attachment
- Series/1 Display Terminal
- Series/1 Disk or Diskette
- Series/1 Printer.

The host system is normally one of the following:

- IBM System/370
- IBM 303x
- IBM 4331 or 4341
- IBM 308x
- IBM 309x.

The Series/1 software requirements, in addition to the basic EDX Version 5 and succeeding supervisor and utilities, consist of the EDX SNA licensed program (5719-XX9).

Support Provided by $RJESNA

The following sections describe the host job entry systems, Series/1 workstation features, and the functions supported by $RJESNA.

Host Job Entry Systems Supported

In order for $RJESNA to communicate with the host, the host operating system must contain a control program, often referred to as a *job entry system.* These job entry systems enable the Series/1 to communicate with the host as a remote workstation by establishing the protocols and SNA session rules to be used while communicating with the host. The job entry systems supported by $RJESNA are:

- OS/VS1 Remote Entry Services (RES)
- OS/VS2 Job Entry Subsystem 2 (JES2)
- OS/VS2 Job Entry Subsystem 3 (JES3)
- DOS/VSE VSE/POWER.

"Host Job Entry System Considerations for $RJESNA" on page 17-4 discusses installation considerations on the host system for the previously mentioned job entry systems.

Workstation Features Supported

The $RJESNA utility supports the Series/1 as an SNA remote job entry workstation with the following features:

- Full function console
- Journaling of session console activity to a data set
- EBCDIC transparency on all data
- Multiple record transmission
- 132-character print line
- Decompression of received data.

Functions of the Workstation

When used as an SNA remote job entry workstation, the Series/1 can perform I/O functions similar to the I/O devices attached locally to the host. Specifically, the Series/1 workstation has the following functions:

- RJE console (Series/1 terminal)
- RJE card reader (Series/1 data set)
- RJE printer
- RJE punch (Series/1 data set).

RJE Console

The console function enables you to send commands to the host or control the operation of the Series/1, such as defining the data sets to be transmitted or defining the print device. The command transmitted to the host consists of up to 70 characters. Messages received from the host can consist of several records up to 120 characters each. Because the typical screen line length is 80 characters, messages greater than 80 characters are divided into two lines to prevent the loss of data.

If the host sends compressed data, the $RJESNA utility performs decompression prior to displaying the data on the console.

When the Series/1 terminal receives "new line" characters (for example, after $RJESNA issues a response to a command you entered), these characters cause the utility to start a new line for the terminal. Data received from the host can contain several new line characters. Multiple new line characters can cause the received message to be rolled off the screen. To avoid the message being rolled off the screen, $RJESNA starts a new line only on the first occurrence of a series of new line characters sent from the host. $RJESNA accepts subsequent new line characters received in a series but ignores them.

RJE Card Reader

The reader function enables you to specify a Series/1 data set to be submitted to the host as a batch job stream. The reader function then transmits this batch job stream to the host. All data to be transmitted to the host is defined as source data; therefore, the reader function creates two 80-character card images from each record in the data set.

The reader function reads data and sends it to the host until one of the following conditions occurs:

- It determines the end-of-file condition by reading the last logical record from the Series/1 data set.
- The utility encounters a /*END control record in the job stream.

RJE Printer

The printer function enables you to receive printed output from a job stream previously submitted to the host. The data records or messages received are 132 characters (maximum) long. If the host sends compressed data, the $RJESNA utility performs decompression prior to printing the data.

The $RJESNA utility assigns the $SYSPRTR device as the default for all output directed to the Series/1 printer. You can specify a printer other than $SYSPRTR; however, you must define that printer at system generation with the TERMINAL configuration statement. $RJESNA does not support special forms.

If you define $SYSPRTR or the printer defined to $RJESNA as a spoolable device and the spool facility is active, the printed output received by $RJESNA is spooled. The output stream from each job is spooled as it is received. For example, if three job streams are submitted to the host system, the output received creates three output streams in the spool data set. The $RJESNA utility attempts to supply the job name, forms name, and number of copies in the spool control record for a print stream if the spool facility is active. However, some host systems supply this information *after* sending the print stream separator pages. In this case, $RJESNA cannot supply this information because the spool control record was already created when $RJESNA encountered the first separator page for that job. If this is not

acceptable for your application, you can define your remote workstation (at the host) so that no separators are sent, or there may be a host command you can send (with the COMMAND command) to "turn separators off." See "Host Job Entry System Considerations for $RJESNA" on page 17-4 or the appropriate host job entry commands manual for information on how to turn separators off.

If the host requests multiple copies of an output data set, $RJESNA creates a single spool job for the data specifying the number of copies. The spool facility prints the output the requested number of times. If the host requests multiple copies and the spool facility is not active, the spool facility prints one copy of the received output.

RJE Punch

The punch function enables you to receive punched output created by a job stream previously submitted to the host. The output data received from the host is placed in the Series/1 data set that you specify as the punch data set. The data received is stored as object or source, depending on how you specify the punch data to be stored.

If you specify the data received as source data, the punch function writes the data to the Series/1 data set, two 80-character card images per record. If you specify the data received as object data, the punch function writes the data to the Series/1 data set three 80-character card images per record.

If the host system compressed the data, the $RJESNA utility decompresses the data before writing it to the data set.

When the punch function writes the output data to the data set specified, it overwrites any previously existing data in that data set. You should ensure that you do not need the data existing in the data set.

Note that the typical host job entry system will, by default, create extra punched output records at the beginning, and ,in some cases, end of each punched output stream. These records are called *separator records*, and the operator uses them to determine the beginning and end of punched streams when the output is actually punched to cards. In the EDX SNA RJE environment, these separator records are not necessary but because $RJESNA cannot differentiate between separators and data, the separator, if received, is written to the punched data set. You may find these separators undesirable if you expect to receive object output to a data set for later $EDXLINK or $UPDATE processing; therefore, you should turn off the separators for punched output, if possible, by either (1) an operator command or (2) defining the workstation punch to not receive separators.

Some of the host job entry systems also create an extra blank punched output record at the end of each punched output stream. The EDX SNA RJE environment does not require this blank record environment, but because $RJESNA does not know if the blank record is part of the user's data or the blank card sent by the host entry system, it writes it to the punched data set. For this reason, you may need to allocate the punch data set one record longer than normal.

Priority of Workstation Functions

Each of the workstation functions is implemented as a separate Event Driven Executive task. Because of this, it is possible for you to issue a function with a high priority and temporarily suspend a function with a lower priority. The following list shows workstation functions in decreasing priority order:

1. RJE console
2. RJE card reader
3. RJE printer/RJE punch (same priority).

It is possible for you to interrupt a print stream being received to submit a batch job stream, then resume printing (in that order). It is also possible to have two suspended functions at one time. For example, you initiate a job submission while the workstation is printing and then interrupt that job submission to send a command to the host. After the command goes to the host, $RJESNA resumes transmitting the batch job. After the transmission completes, the printing resumes.

The $RJESNA utility can always suspend a transmission *to* the host, but $RJESNA can only request the suspension of a transmission *from* the host. It is the host's option to grant or deny the request for suspension. If the host denies the request for suspension, the Series/1 workstation continues to receive and process the transmission from the host. $RJESNA continues to request suspension until the host grants the request or the transmission from the host ends. When either of these events occur, $RJESNA processes the function with the highest priority.

How $RJESNA Processes Job Stream Control Records

When you create job streams to be submitted to the host for processing, you have the option to include a /*END or /*CONCAT control record. This section describes how the utility processes these control records.

/*END Record

If $RJESNA encounters the characters /*END in the first five positions of a card image in the job stream, it sets an end-of-file condition and terminates processing of that job stream. The /*END record indicates the last record of the job stream and is not transmitted to the host.

/*CONCAT Record

If $RJESNA encounters the characters /*CONCAT in the first eight positions of a card image, it scans the rest of the card image for a data set name and a volume name separated by a comma. When it finds the data set specified on the /*CONCAT record, it closes the original data set and opens the new data set, and transmission continues. If an error occurs in the data set or volume name specified on the /*CONCAT record (such as the volume not being online, specified incorrectly, or the data set not found), $RJESNA displays an appropriate message on your terminal and terminates the function. $RJESNA cancels all previously transmitted data in the job stream to prevent the host from executing a partially submitted job stream. The utility does not transmit the /*CONCAT record to the host.

When $RJESNA completes processing the concatenated data set, $RJESNA does *not* return to the previous data set to finish processing it. The /*CONCAT record should be the last record in a data set to be transmitted.

There is no limit to the number of data sets that can be concatenated.

Starting $RJESNA

You start the $RJESNA utility by pressing the Attention key on your terminal and entering $L $RJESNA. $RJESNA determines whether the EDX SNA is already active. If SNA is not active, $RJESNA prompts you for the network address (XID) of your Series/1 workstation and then loads SNA. The $RJESNA utility assumes that SNA resides on the same volume from which you loaded $RJESNA. If $RJESNA encounters an error while attempting to load SNA, it displays an error message on your terminal indicating the error. SNA may, but is not required to, run in the same partition in which $RJESNA is loaded.

After SNA is loaded (or already active), $RJESNA prompts you for the following information required to complete the host connection:

- The name of the host application (for example, JES2) as defined by the host systems programmer
- The name of the remote workstation ID.

The $RJESNA utility creates the host logon request (INIT-SELF) from this information and transmits it to the host subsystem. If the logon is successful, $RJESNA issues a message indicating it is ready for work. If the logon is unsuccessful, $RJESNA returns an error message to you and terminates. Once successful initiation completes, you can issue the $RJESNA operator commands. These operator commands are described in "$RJESNA Operator Commands" on page 16-8.

In the event the remote workstation ID requires a password, a host line password, or other required parameters, you should supply this information when you answer the prompt for the remote ID. For JES2, the logon format is:

```
RMTID,LINEPSWD,IDPSWD
```

When answering the prompt for $RJESNA and supplying an ID password, the format is:

```
RMTxx,,PSWD
```

For RES, the logon format is:

```
RMTID/PSWD TERM(#)
```

You should refer to the appropriate reference manual for the host entry system you are using for additional information on the correct format for entering the workstation ID.

Note: The maximum length of the response to the workstation ID prompt is 20 characters (including commas or blanks).

You can start $RJESNA also through the $JOBUTIL utility. Instead of being prompted as above, you can supply the required host application name and remote workstation ID on the PARM card. If you do not specify either of these parameters on the PARM card, $RJESNA prompts you for this information. The control cards needed to start $RJESNA through the $JOBUTIL utility are as follows:

```
JOB     jobname
PROGRAM $RJESNA,volume name
PARM    aaaaaaaa,bbbb...b
EXEC
EOJ
```

Where:

aaaaaaaa is the host application name (1 – 8 characters)

bbbb...b is the remote workstation ID (1 – 20 characters).

Examples

The following are two typical examples of starting $RJESNA through the $JOBUTIL utility.

Example 1:

This example shows the host application name, the remote workstation ID, and the host ID password specified on the PARM card.

```
JOB     JOB1
PROGRAM $RJESNA,EDX003
PARM    JES2,RMT24,,PASS1
EXEC
EOJ
```

Example 2:

This example shows only the host application name and the remote workstation ID specified on the PARM card. In this example, $RJESNA resides on the IPL volume.

```
JOB     JOB2
PROGRAM $RJESNA
PARM    JES2,RMT25
EXEC
EOJ
```

Note: When you use $JOBUTIL to start $RJESNA, use the information preceding the first comma for the name of the host application. Use subsequent characters (up to 20) to supply the remote workstation ID and passwords if you defined passwords for logon.

$RJESNA Operator Commands

This section describes the operator commands for the $RJESNA utility. The following is a brief description of the commands and their function followed by a more detailed description of each command:

ABORT Terminates the current data transmission to or from the host

COMMAND
Prompts you for a control command or information request to be sent to the host

ENDRJE Terminates the remote job entry utility

HRJE Displays the supported $RJESNA operator commands

JOURNAL
Logs $RJESNA messages and operator commands to a Series/1 data set

PRINTON
Defines the device name to receive printed output

PUNCHO
Defines the disk or diskette data set to receive punched object data output

PUNCHS Defines the disk or diskette data set to receive punched source data output

RESET Resets the punch data set name and format

SUBMIT Defines and transmits a data set to the host

SUBMITX
Defines and transmits a data set to the host.

You enter these operator commands by first pressing the Attention key and then entering the particular command.

ABORT

The ABORT command terminates a data transmission that is currently in process. Upon receipt of the ABORT command, the utility resets the function to an inactive but ready state.

An ABORT command issued during a SUBMIT or SUBMITX operation causes SNA to send the "abort data sequence" command to the host, preventing the submitted job stream from executing.

An ABORT command issued while the Series/1 workstation is receiving a print or punch data stream causes SNA to send an "abort data sequence" command the host. The host, in most cases, aborts the operation and purges the data stream.

The $RJESNA abort sequence terminates the transmission and, depending on the host RJE system, terminates (deletes from the queue), suspends (requiring a host command intervention for cancel or restart), or retries (the host restarts the transmission) the data transmission.

Note: Because not all job entry systems respond to the ABORT command in the same manner, the recommended way to abort a data stream being received from the host is to issue the appropriate host cancel command as described in the manuals for the particular job entry system being used.

COMMAND

The COMMAND command sends a single entry system command to the host. The most common use of COMMAND is to send control commands and information requests to the host.

The $RJESNA utility does not require sending a /* prefix on the command. You can include the /* prefix, however, but $RJESNA removes the /* before transmitting the data.

Upon entering the COMMAND command, you can specify (on the same line) the command to be sent. If you do not specify the command on the same line, $RJESNA prompts you for this information. For example:

```
> COMMAND $DQ
    or
> COMMAND

  ENTER COMMAND: $DQ
```

When the utility receives the command, it sends the data to the host. The utility performs actual transmission of the data when the host is in a receive state.

ENDRJE

The ENDRJE command terminates the $RJESNA utility, and you should use it to terminate sessions with all host entry systems *except* DOS/VSE POWER. For information on session termination with DOS/VSE POWER, see "Terminating $RJESNA" on page 16-14.

Issue ENDRJE when all processing is completed and you wish to terminate the session with the host.

When the $RJESNA utility receives the ENDRJE command, the session and the utility terminate after the completion of all current activity. $RJESNA sends an SNA Request Shutdown sequence to the host when the current activity completes and then terminates. The ENDRJE command is discussed further in "Terminating $RJESNA" on page 16-14.

HRJE

The HRJE command displays the supported $RJESNA operator commands on your terminal. To receive the list of commands, enter the HRJE command and press the Enter key on your terminal.

JOURNAL

The JOURNAL command keeps a record of all console activity during a $RJESNA session. This command causes $RJESNA to write all data written by $RJESNA and all $RJESNA commands you entered during a session to a data set.

Before you issue the JOURNAL command, you must allocate a data set with the $DISKUT1 utility. The data set size depends on how much activity you expect during a session. The records written to the data set are in EDX source format to allow you to browse the data set using the $FSEDIT utility or print the data set using the $DISKUT2 utility. $RJESNA writes the data set and when it encounters the physical end-of-file, it wraps the data to the beginning of the data set (record 2). This wrapping technique allows $RJESNA to write to the data set continuously without losing the latest data.

Record 1 is a control record with an ID of **$RJESNA**. This control record contains the number of console lines recorded since the last wrap or since the opening of the data set (if never wrapped). In addition, this record contains the number of times the data set was wrapped.

To use the JOURNAL facilities, you must enter the command JOURNAL ON followed by the data set name and volume name separated by a comma. For example:

```
> JOURNAL ON data set,volume
```

You can also enter just the JOURNAL command and be prompted for the **ON** or **OFF** operand, and subsequently, the data set name and volume name.

$RJESNA prompts you for the data set name and volume name if you do not specify them on the command. The data set name is a required parameter. If you enter null characters or blanks in response to the prompt, $RJESNA reprompts you. If you do not specify the volume name, $RJESNA assumes the data set resides on the IPL volume. In the event $RJESNA cannot find the data set or encounters an error when opening the data set, it issues an appropriate error message and JOURNAL terminates.

To terminate the JOURNAL function, you can either enter the command JOURNAL OFF (parameters not required) or allow $RJESNA to terminate the function when the program terminates. If, during termination of the JOURNAL function, an odd number of console lines have been journaled, $RJESNA pads the last physical EDX record written to the data set with a blank console line (80 bytes). This is due to the EDX format requirements for source data sets—two 80-character logical records per physical record.

The JOURNAL function is optional. You can exclude it during $RJESNA installation, saving execution storage. This procedure is described in the Program Directory.

PRINTON

The PRINTON command defines the name of the output device to receive printer output from the host. The name can be a 1–8 character name of a printer assigned to the spool facility. $RJESNA uses $SYSPRTR as the default printer until the first PRINTON command is issued.

Upon entering the PRINTON command, you can specify (on the same line) the printer name. If you do not specify the printer name on the same line, $RJESNA prompts you for this information. For example:

```
> PRINTON PRTR1
```

or

```
> PRINTON

  ENTER PRINTER NAME: PRTR1
```

If you do not specify the device name (null characters or blanks are entered), $RJESNA reprompts you. The device name you specify remains in effect until you change it by issuing another PRINTON command. $RJESNA does not accept a PRINTON command issued while the printer is printing.

If you enter an invalid device name (such as a device not defined by a TERMINAL statement at system generation), $RJESNA displays subsequent print output received at the Series/1 on the device from which you loaded $RJESNA.

PUNCHO and PUNCHS

The PUNCHO and PUNCHS commands define the disk or diskette data set to be used to receive punched output from the host. Card image punch data streams can be written to disk or diskette in two different formats: source or object. The PUNCHS command specifies source data, whereas the PUNCHO command specifies object data. Source format produces two 80-byte card image records per 256-byte disk record, with the second card beginning at byte location 129. Object format produces three 80-byte contiguous card image records per 256-byte disk record, with the last 16 bytes set to hexadecimal zeros. The punch specification (data set name and type) clears (set to blanks) at the completion of each punch data stream so that separate punch data streams are not stored into the same EDX data set.

Upon entering the PUNCHO or PUNCHS command, you can specify (on the same line) the data set name and volume name to be used for punched output. If you do not specify the data set name and volume name on the same line, $RJESNA prompts you for this information. For example:

```
> PUNCHS PUNCHOUT,WRKLIB
```

or,

```
> PUNCHS

  ENTER PUNCH FILE NAME (NAME,VOLUME): PUNCHOUT,WRKLIB
```

If you do not specify the volume name, $RJESNA assumes the IPL volume.

The utility does not test the existence of the punched output data set until $RJESNA attempts to write to it. If $RJESNA cannot find the data set or an error occurs opening the data set, $RJESNA displays an appropriate message at your terminal.

RESET

The RESET command resets (sets to blanks) the punch data set name (defined by the last PUNCHO or PUNCHS command) and format (O-object, S-source) if punching has not yet started.

The following is an example of the RESET command:

```
RESET PU
```

If $RJESNA does not enter PU, it prompts you to verify the reset of the punch data set name and format.

SUBMIT and SUBMITX

The SUBMIT and SUBMITX commands transmit a data stream from Series/1 disk or diskette storage to the host. Because the $RJESNA utility transmits transparent data only, the functions of the SUBMIT and SUBMITX commands are identical. You can send multiple disk or diskette data sets using the /*CONCAT statement in the data stream itself. The data streams must be in the same format as that produced by the $EDIT1N and $FSEDIT utility programs (for example, two 80-byte card image records per 256-byte disk or diskette record with the second card beginning at byte location 129).

$RJESNA recognizes two command statements within the data stream, which are not transmitted to the host:

/*END Indicates the end of the data stream to be sent

/*CONCAT
Indicates the data stream is to be continued using the data set and volume name specified. If you do not specify the volume, $RJESNA assumes the IPL volume. Any number of data sets can be concatenated into one data stream.

Note: The $RJESNA utility cannot detect "concat loops." This condition can occur if you submitted a job stream with nested /*CONCAT records, as illustrated in the following:

```
MYJOB1,WRKLIB

    //JOBA JOB (ACCOUNT),USER,TIME=1
        •
        •
        •
    /*CONCAT   MYJOB2,WRKLIB

MYJOB2,WRKLIB

    //JOBB JOB (ACCOUNT),USER,TIME=1
        •
        •
        •
    /*CONCAT   MYJOB3,WRKLIB

MYJOB3,WRKLIB

    //JOBC JOB (ACCOUNT),USER,TIME=1
        •
        •
        •
    /*CONCAT   MYJOB1,WRKLIB   loop begins here
```

The utility remains in the submit loop until you either enter an ABORT command or a $C command to cancel $RJESNA. The ABORT command is the preferred method of terminating this type of loop. The ENDRJE command, if issued, will not work.

Upon entering the SUBMIT or SUBMITX command, you can specify (on the same line) the data set name and volume name to be transmitted to the host. If you do not specify the data set name and volume name on the same line, $RJESNA prompts you for this information. For example:

```
> SUBMITX MYJOB,WRKLIB
```

or

```
> SUBMIT

 ENTER SUBMIT FILE NAME (NAME,VOLUME): MYJOB,WRKLIB
```

If you do not specify the volume name, $RJESNA assumes the IPL volume. If $RJESNA cannot find the data set and/or volume name, $RJESNA terminates the command and issues a message describing the error.

Terminating $RJESNA

After you have established a session with the host and you no longer require host processing during the session, you can terminate $RJESNA by entering either the ENDRJE command or the appropriate host entry system termination command.

The difference in the two methods has to do with the origin of the session termination request. When the ENDRJE command executes, $RJESNA requests session termination at the completion of all current activity. When a host entry system session termination command is issued, the host requests session termination.

Because $RJESNA does no processing of a host session termination command, the session may be terminated *immediately* or at the completion of current activity. The host session termination command can result in the termination of the session in the middle of the Series/1 receiving a print or punch output stream.

When the host entry system requests session termination, the message **$RJESNA SESSION TERMINATED BY HOST**, along with any other termination messages, appears at the terminal from which you loaded $RJESNA.

When operating with DOS/VSE POWER, the ENDRJE command is not valid and causes unpredictable results if issued. To terminate a DOS/VSE POWER session, you should use the POWER command **SIGNOFF**. This command causes the host to terminate the session when all current activity completes.

$RJESNA Error Handling

When the EDX or SNA encounters an error, $RJESNA handles the error as follows:

- $RJESNA issues no message if the EDX has already presented the error.
- If $RJESNA receives a return code indicating an error and the error was not previously reported, $RJESNA issues the appropriate error message. These errors include EDX errors as well as SNA error conditions.
- $RJESNA presents any other errors encountered in a message to you.

The messages issued by $RJESNA are described in Appendix A, "Messages."

$RJESNA Task Error Exit

The $RJESNA utility program includes a task error exit routine. If EDX passes control to this routine, the routine issues a message containing the current level status block (LSB), active tack control block (TCB) address, and the contents of that TCB at the time $RJESNA encountered the error. This message appears at the terminal from which you loaded $RJESNA. The message may appear on the current $RJESNA printer if $RJESNA was printing at the time of the error. After issuing the message, $RJESNA stops execution.

If you wish to extend the logic of the task error exit, you can write your own exit routine and subsequently replace the routine provided by $RJESNA with your own routine. If you write your own error exit routine, the routine must contain two entry points, **CDSTERRX** and **CDSTERRD**. You must name the routine you provide **CDSTERRX**. **CDSTERRD** is the label of a 12-word data storage area.

Refer to the *Language Programming Guide* or the *Customization Guide* for additional information on writing a task error exit routine.

Adding Your Own Task Error Exit to $RJESNA

To replace the $RJESNA task error exit routine and add your own to $RJESNA, you must perform the following:

1. Assemble your EDL task error exit routine, create a data set with the name **CDSTERRX**, and place the object output in that data set. This data set can reside on any volume.
2. Edit the INCLUDE statement in your copy of the $RJESNA link control data set to reflect the name of the volume your routine resides on. The INCLUDE statement to be edited has the name CDSTERRX specified on it.
3. Relink ($EDXLINK) $RJESNA using the updated link control data set as described in the Program Directory for $RJESNA.

Sample $RJESNA Session

Figure 16-1 is a sample session with $RJESNA. Information or operator commands that you enter are in boldface. The character " > " appears in the sample to indicate where you press the attention key to execute an $RJESNA operator command.

```
> $L $RJESNA
$RJESNA     nnP, hh:mm:ss, LP= nnnn

ENTER HOST ID: JES2
ENTER REMOTE ID: RMT01
HOST CONNECTION ESTABLISHED
FOR HELP ENTER ATTN "HRJE"

> COMMAND

ENTER COMMAND: /*$DA      (/* prefix optional)

COMMAND READY TO SEND
COMMAND SENT
 HOST RESPONSE LINE 1
 HOST RESPONSE LINE 2
        •
        •
        •
> PUNCHO

ENTER PUNCH FILE NAME (NAME,VOLUME): PCHOUT1,EDX002
PUNCH FILE DEFINED

> SUBMIT

ENTER SUBMIT FILE NAME (NAME,VOLUME): RJEJOB01,EDX002
```

Figure 16-1. Sample $RJESNA Session with Host

An example of the job stream (RJEJOB01) might appear as follows:

```
          //JOBA JOB (ACCOUNT),USER,TIME=1
          //STEP1 EXEC ASMFCLG
          //ASM.SYSIN  DD  DSN=A.B.C(D),
          //           DISP=SHR
SUBMIT FILE READY TO SEND
FILE TRANSMISSION STARTED
FILE TRANSMISSION COMPLETED

> PRINTON

ENTER PRINTER NAME: PRTR1
PRTR1 DEFINED AS $RJESNA PRINTER
> COMMAND $DQ    (/* not used)

COMMAND READY TO SEND
COMMAND SENT
 HOST RESPONSE LINE 1
 HOST RESPONSE LINE 2 (etc.)
         •
         •
         •
```

$RJESNA identifies a punch stream being received. The punch data received is placed in the data set defined by the previous PUNCHO command to data set PCHOUT01,EDX002:

```
PUNCHING STARTED
PUNCHING COMPLETED
NUMBER OF CARDS RECEIVED IS  150
NUMBER OF RECORDS WRITTEN TO FILE PCHOUT01,EDX002 IS 50

> SUBMIT RJEJOB02

SUBMIT FILE READY TO SEND
FILE TRANSMISSION STARTED
FILE TRANSMISSION COMPLETED
PUNCH DATA BEING RECEIVED - NO PUNCH FILE DEFINED
ENTER PUNCH FILE NAME (NAME,VOLUME): PCHOUT02,EDX002
ENTER PUNCH FORMAT - S OR O: S

PUNCHING STARTED
PUNCHING COMPLETED
NUMBER OF CARDS RECEIVED IS  202
NUMBER OF RECORDS WRITTEN TO FILE PCHOUT02,EDX002  101

> ENDRJE

$RJESNA PROCESSING COMPLETED
$RJESNA ENDED AT hh:mm:ss
```

Figure 16-2. Job Stream Example (Sample $RJESNA Session with Host)

Chapter 17. Installing the Remote Job Entry Utility

This section outlines the installation steps required to use the $RJESNA utility. See "Host Job Entry System Considerations for $RJESNA" on page 17-4 for the requirements for the host system, and Chapter 2, "Installing and Defining the SNA Network for Use with SDLC" for details on SNA installation with SDLC support, or Chapter 3, "Installing and Defining the SNA Network for Use with Shared SDLC" for details on SNA installation with extended data link manager support.

Installing the remote job entry support for the $RJESNA utility requires you to perform several steps. These installation steps can be classified as host-related and Series/1-related.

Briefly, the host-related steps involve defining the host job entry subsystem, protocols, and defining the Series/1 to the SNA network.

The Series/1-related steps, in addition to system generation, require you to define spool data sets if spooling is used, and to install the Event Driven Executive SNA support.

Host-Related Steps

To establish the environment for $RJESNA on the host system, you must perform the following steps:

1. Define the host job entry subsystem to the host access method.

   Include the job entry subsystem information in the host VTAM definitions or the TCAM Message Control Program.

2. Define the remote workstation(s) to the host job entry subsystem.

3. Describe the protocols to be used in talking between the host and the remote workstation.

   Include the appropriate session parameters in the local VTAM definitions (Modetab generation).

4. Define the Series/1 to the network.

   Include the Series/1 cluster definition in the 3705 Network Control Program (NCP) system generation.

Series/1-Related Steps

To establish the environment for $RJESNA on the Series/1, you must perform following steps:

1. Install EDX on the Series/1. Refer to the *Language Programming Guide* or the *Installation and System Generation Guide.*

2. Define the spool data set if you use spool. Refer to the *Operator Commands and Utilities Reference Summary.*

3 Install SNA on the Series/1. See Chapter 2, "Installing and Defining the SNA Network for Use with SDLC," or Chapter 3, "Installing and Defining the SNA Network for Use with Shared SDLC."

The actual $RJESNA installation consists of:

1 Installing as directed in the installation material supplied with $RJESNA

2 Defining any data sets to be used for submitting job streams, receiving punched output streams, or journaling.

Note: If $RJESNA is to load the Event Driven Executive SNA, both program products must reside on the same volume.

The $RJESNA utility requires a minimum of 22K bytes of Series/1 storage to execute. This storage requirement includes $RJESNA and the Series/1 SNA $NETCMD module (application link to Series/1 SNA).

Receiving Data from the Host

The host system has two methods available to it for effective data transmission to the Series/1. These methods are called data *compression* and data *compaction.*

Data Compression

Data compression is a technique where strings of duplicate characters are substituted with a 1- or 2-byte code and trailing blanks are truncated before the data is transmitted. These duplicate characters and trailing blanks are reinserted (decompressed) when the data is received.

Data Compaction

Data compaction involves restructuring portions of the data stream so that some bytes represent more than one character of data. When you use compaction, you are responsible for selecting the characters to be compacted. When the compacted data is received, it is put back in its original state (decompacted).

The $RJESNA utility performs decompression functions but does not perform decompaction.

Performance Considerations

For certain types of data transmissions, data compression or compaction can significantly reduce the number of characters in a data stream transmission, thus saving transmission time. Compaction, however, requires the use of computation time to save transmission time. Compaction may not be efficient when the data to be transmitted has only a few occurrences of character strings that can be compacted.

Considerations for Writing Decompaction Routines

The $RJESNA utility has two WXTRN (weak external reference) type addresses that enable you to supply a routine to perform decompaction of incoming data if required. The two addresses refer to your decompaction routine and the address of the output buffer for decompacted data, respectively. If you supply this routine, you

must name it **DPACTRJE** (first WXTRN), and you must name the output buffer **DPACTBUF** (second WXTRN).

When $RJESNA encounters decompaction-related data and you have supplied a decompaction routine, $RJESNA calls the routine as a standard Event Driven Language subroutine with two parameters.

The first parameter is the address of a buffer containing either:

- Function Management 2 or 3 header, indicating the compaction table
- Up to 64 bytes of data, starting with a string control byte (SCB) to be decompacted.

The second parameter is a single precision variable containing either:

- A zero (if the first parameter is the address of a buffer containing a Function Management header)
- The length, in bytes, of the data to be decompacted whose address is supplied as the first parameter.

Therefore, in order for you to determine how to process the first parameter, you must test the contents of the second parameter.

If you supply a Function Management header, you must save (in your subroutine's data area) the compaction table supplied. Subsequent calls to DPACTRJE should be data to decompact using that table until you specify a new table.

After handling the data or Function Management header supplied, your routine must move a single precision return code into the second parameter. A value of -1 indicates success, while *any* value other than −1 indicates failure. If $RJESNA receives a failing return code, $RJESNA rejects the SNA request unit (RU); this can result in aborting the data stream.

If the subroutine decompacted data when it was called, you must place the resulting output into a data area in DPACTRJE labeled **DPACTBUF** (minimum size of 130 bytes).

The first word of DPACTBUF must contain the length, in bytes, of the decompacted data which follows it (starting in the second word of DPACTBUF). This length does *not* include the length of the first word.

The DPACTRJE routine must link with the $RJESNA program to be available for decompaction. $RJESNA provides link instructions and a link control data set with the installation material.

The host systems programmer must change the job entry system initialization parameters on the host to facilitate the use of compaction. The information in "Host Job Entry System Considerations for $RJESNA" on page 17-4 identifies the parameters relating to compaction and defines these parameters for a noncompaction environment only.

If $RJESNA receives compacted data and a decompaction routine does not exist, $RJESNA rejects the data with a "function not supported" (1003) sense code to the host and issues a message.

Requirements for Decompaction Routines

The following are the specific coding requirements for your decompaction routine:

- A SUBROUT statement with the name of DPACTRJE and two parameters. For example:

```
SUBROUT   DPACTRJE,parm1,parm2
```

- A data area of at least 130 bytes, labeled DPACTBUF. It must be aligned on a word boundary. For example:

```
DPACTBUF  DATA   65F'0'
```

- An ENTRY statement for both DPACTRJE and DPACTBUF. For example:

```
ENTRY     DPACTRJE,DPACTBUF
```

- A RETURN statement to return control to $RJESNA.
- A data area large enough to store a host decompaction table.
- Do not use any instruction that may cause the calling $RJESNA task to enter a wait state (such as ENQ, DEQ, ENQT, DEQT, ATTACH, DETACH, LOAD, WAIT, READ, WRITE, STIMER, or TP).
- Do not use the ENDTASK and PROGSTOP instructions.
- Do not alter the contents of the first parameter; it is an address.
- If you require any entry points other than DPACTRJE or DPACTBUF, ensure the labels you use do not duplicate any $RJESNA entry points. To ensure this does not happen, install $RJESNA without your decompaction routine and examine the entry points of the $RJESNA program from the link storage map.

Host Job Entry System Considerations for $RJESNA

This section describes the installation parameters required by the supported host job entry systems to communicate with SNA and $RJESNA. We discuss only those parameters applicable to remote job entry or in some other way important to RJE. For many of the parameters, we discuss specific operands and values. Define those specific operands and values in the host entry system generation to ensure proper communication between the host and the Series/1. Implement parameters not defined or discussed at your discretion.

Defining $RJESNA for VTAM

This section contains information relevant to the definition and generation of the VTAM network control program (NCP) for $RJESNA.

Network Control Program Generation

$RJESNA uses NCP macros and operands in two ways. First, the macros and operands generate the NCP load modules. Second, VTAM uses them to obtain information about the SNA network.

The following NCP macro and operands define $RJESNA workstations:

```
         PU    ADDR=chars
               ,MAXDATA=size
               ,MAXOUT=count
               ,PASSLIM=n
               ,PUTYPE=2

label    LU    ,LOCADDR=n
               ,PACING=(n,m)
```

Notes:

1. Define one PU for each Series/1.
2. Define one LU for each workstation.

VTAM-only operands:

```
         PU    ISTATUS=
               VPACING=
         LU    MODETAB=
               BUFLIM=
```

The PUTYPE operand of the PU macro must identify each Series/1 as a type 2 Physical Unit. The MODETAB operand of the LU macro should specify the name of a logon mode table that provides the BIND parameters when a workstation LU logs on. Refer to *ACF for NCP and System Support Programs Installation and Resource Definition* for details on the above macros and for specifying the appropriate NCP level.

You must use the pacing values specified on the LU macro to determine the number of SNA send and receive buffers ((2 x n) - m) to be defined to Series/1 Event Driven Executive SNA. For example:

Pacing Count	Buffers
2,1	3
7,1	13
X,1	2X-1

The pacing count 2, 7, or X is the number of data buffers Series/1 SNA can receive before it sends a pacing response. The value 1 indicates that the first buffer (of the 2 or 7) of data from the host system should request the pacing response from Series/1 SNA.

See Appendix E, "Node Definition Statements" for more information on NCP installation.

VTAM Definitions

When you add support for $RJESNA to the VTAM system, it may affect the VTAM APPL statement if no other SNA RJE workstations are currently in use.

You must code the APPL statement to define the host entry system to VTAM (for example, JES2). The following is an sample definition for JES2:

```
JES2     APPL  ACBNAME=JESACB,                                  C
               BUFFACT=
(see NCP definitions for value),
               PRTCT=JESPSWD
```

The ACBNAME value must correspond to the name specified for the following:

- JES2, in the APPLID parameter of the LOGONn initialization statement
- JES3, the APPLID parameter of the COMMDEFN initialization statement
- RES, the APPLID parameter of the RTAM macro instruction
- DOS/VSE, the APPLID defined on the SNA operand of the POWER macro.

The PRTCT (password) value must correspond to the value specified for the following:

- JES2, the PASSWORD= operand of the LOGONn initialization parameter
- JES3, the P= operand of the COMMDEFN initialization statement
- RES, the PASSWD= operand of the RTAM macro instruction
- DOS/VSE, the password defined on the SNA operand of the POWER macro.

The mode table entry for $RJESNA can be an IBM-supplied mode table entry or an entry constructed by assembling a series of MODETAB, MODEENT, and MODEEND macros and linking the output into the SYS1.LPALIB data set. The mode table entry defines the BIND image transmitted to the $RJESNA work station. We explain the variations in the mode table entry or the method used by the host to construct the table entry, as they apply to various host entry systems, in the following paragraphs.

The $RJESNA utility as distributed by IBM contains a mode table entry with the name of **BATCHS1**. You may either define a mode table entry as specified in the following host subsystem descriptions and specify it as BATCHS1 or change the default to an existing mode table entry name. If you wish to modify the mode table entry name in the $RJESNA utility, you can patch $RJESNA after installation.

Use the $DISKUT2 utility to patch the four words at location LUHOST+2. You can find the entry point for LUHOST in the $RJESNA link storage map.

See Appendix C, "VTAM Considerations" for further information on VTAM installation.

Define the mode table name in the NCP PU macro. Define the mode table entry name in the **LOGMODE=name** parameter of the MODETAB macro.

Defining $RJESNA for OS/VS2 MVS (JES2)

When JES2 starts, it processes a series of initialization parameters to determine the configuration options to be used. Initialization parameters required in order to use $RJESNA can be characterized as system-wide initialization parameters and remote workstation definitions.

The system-wide initialization parameter for SNA/SDLC or remote job entry support for JES2 is:

- LOGONn — Specifies JES2 as a VTAM application program and identifies the ACB name and password to be specified on the VTAM APPL statement.

The applicable remote workstation parameters for SNA/SDLC definition for JES2 are:

- LINEnnn — Specifies the characteristics of one logical line to be used for telecommunications during remote job entry. You must define one line for each $RJESNA workstation that is to be concurrently active. The definition must specify UNIT=SNA and may specify a line password.
- RMTnnn — Specifies the characteristics of one remote workstation. The parameter operands are:

  Terminal type
  : Must specify as **LUTYPE1**.

  Bufsize=nnn
  : Should be **256**.

  COMP|NOCOMP
  : Identifies whether compression is to be used; should specify **COMP**.

  CONSOLE|NOCON
  : Indicates the presence of a separate console function at the workstation. Specify **CONSOLE**.

  CMPCT|NOCMPCT
  : Specifies whether compaction is to be used for this workstation. Specify **NOCMPCT**.

  FIXED|VARIABLE
  : Specifies variable or fixed data record length. Specify **VARIABLE**.

  NUMPR=n
  : Specifies the number of logical printers at this workstation. Specify **1**.

  NUMPU=n
  : Specifies the number of logical punches at this workstation. Specify **1**.

NUMRD=n
Specifies the number of logical readers at this workstation. Specify **1**.

PASSWORD=cccccccc
Specifies the password to be used by this workstation when it logs on.

SETUPHDR|SETUPMSG
SETUPHDR specifies that a peripheral data information record (PDIR — FM header type 2), containing setup information, is sent to the workstation. The workstation then processes the PDIR. SETUPMSG specifies JES2 is to generate any required operator messages instead of sending PDIRs to the workstation. Specify **SETUPHDR**.

SETUPINF|SETUPACT
SETUPINF specifies that operator setup messages at the host console are deleted automatically. SETUPACT specifies that operator setup messages require specific operator action to delete them. Specify **SETUPINF**.

WAITIME=nnn
Specifies the length of time JES2 waits after processing an input or output stream to allow entry of commands or jobs by the RJE operator. Specify **1 second**.

- Rnnn.PR1 — Specifies the characteristics of the logical printer function at a remote workstation. *nnn* specifies the number of this workstation. The following are additional operands for this definition:

AUTO|OPERATOR
Specifies forms change mode. Specify **OPERATOR**.

CCTL|NOCCTL
Specifies carriage control characters and SNA Character Strings (SCS) are to be placed in the output stream for the printer. You must specify **CCTL**.

CHAINSIZ=nnnnn
Specifies the maximum number of print lines that constitute a chain. The value should reflect the value that constitutes a page or should not be specified.

CMPCT|NOCMPCT
Specifies whether compaction is to be used. Specify **NOCMPCT**.

COMP|NOCOMP
Specifies whether compression is to be used. Specify **COMP**.

COMPACT=n
Specifies the compaction table to be used. Specify **COMPACT=0**.

FCBLOAD|NOFCBLOD
Specifies whether JES2 is to build the forms control block for the workstation. Specify **NOFCBLOD**.

LRECL=nnnn
Identifies the logical record length of the print line. Specify **132**.

NOSEP|SEP
: Specifies whether or not JES2 is to provide separator pages between jobs.

PRWIDTH=nnn
: Specifies the maximum number of characters to be printed on one line. Specify **132**.

SELECT=PRINTn|EXCHn|BASICn
: Specifies the device type to which queued output for this printer is sent. PRINT means the output is for the printer; n specifies the printer number, which should be 1. Specify **PRINT1**.

- Rnnn.PU1 — Specifies the characteristics of the logical punch function at a remote workstation; *nnn* specifies the number of this workstation. The following are additional operands for this definition:

  AUTO|OPERATOR
  : Specifies forms change mode. Specify **AUTO**.

  CCTL|NOCCTL
  : Specifies placing of carriage control characters and SNA Character Strings (SCS) in the output stream for the punch. Must specify **CCTL**.

  CMPCT|NOCMPCT
  : Specifies whether compaction is to be used. Specify **NOCMPCT**.

  COMP|NOCOMP
  : Specifies whether compression is to be used. Specify **COMP**.

  COMPACT=n
  : Specifies the compaction table to be used. Specify **COMPACT=0**.

  LRECL=nnnn
  : Identifies the logical record length of the punch record. Specify **80**.

  NOSEP|SEP
  : Specifies whether or not JES2 is to provide separator pages between jobs.

  SELECT=PUNCHn|EXCHn|BASICn
  : Specifies the device type to which queued output for this punch is sent. PUNCH means the output is for the punch data set; *n* specifies the punch number. Specify **PUNCH1**.

- Rnnn.RD1 — Specifies the characteristics of the logical reader function at a remote workstation; *nnn* specifies the number of this workstation. The following are additional operands for this definition:

  PRLCL|PRRMT
  : Specifies the route code specified by PRDEST as a local printer or a remote workstation printer.

  PUDEST=nnn
  : Specifies the default punch destination for the punched output for all jobs entered at the reader.

  PULCL|PURMT
  : Specifies the route code specified by PUDEST as a local punch or a remote workstation punch.

The only value in the mode table entry that affects JES2 operations is the alternate code value in the COMPROT field. As a result, mode table entries such as the default entry for the IBM 3767 are usable by a workstation. JES2 constructs the BIND image transmitted to $RJESNA workstations using RMTnnn operands.

BIND Image

The BIND image for RJE, as defined by assembling a series of MODETAB, MODENT, and MODEND macros, and link editing the resultant output into the SYS1.VTAMLIB data set, requires the following subparameters:

LOGMODE=name
: Where *name* corresponds to the mode table entry name BATCHS1. You can also modify the name of the mode table entry name, by changing $RJESNA, as described previously in "VTAM Definitions" on page 17-6.

FMPROF=X'03'
: Required.

TSPROF=X'03'
: Required.

PRIPROT=X'A3'
: Specifies multiple request unit chains (required), definite response chains (required), compression (optional), and primary transmission of end bracket.

SECPROT=X'A1'
: Specifies multiple request unit chains (recommended), definite response chains (required), no compression (required), and secondary transmission of end bracket (required).

COMPROT=X'7080'
: Specifies function management headers (required), brackets (required), bracket termination rules (required), ASCII not used (required), and half-duplex flip-flop transmission (required).

RUSIZE=X'8585'
: Specifies a maximum request unit size of 256 bytes. RTAM assumes these values if specified as zero.

PSERVIC=X'01112000B100C00000010040'
: Specifies LU profile 1 (required), function management headers (required), no compaction but peripheral data records is transmitted (optional), use of formatting characters (required), document output (optional), use of transparent record separator characters (default), half-duplex flip-flop protocol (recommended), and card format (optional).

Defining $RJESNA for OS/VS2 MVS (JES3)

JES3 reference texts identify Remote Job Entry (RJE) as Remote Job Processing (RJP). To help eliminate the possible confusion between the JES3 manuals and this section, we use the acronym RJP in the following discussion.

When JES3 starts, it processes a series of initialization parameters to determine the configuration options to be used. These parameters are:

- COMMDEFN — Defines JES3 to VTAM as a VTAM application program. The following parameter applies:

  APPLID
  : Specifies the 1 – 8 alphanumeric character name associated by VTAM as JES3.

- CONSOLE — Specifies the characteristics of an RJP workstation console. Subparameters are:

  JNAME = name
  : Specifies the name of the workstation with which this console is associated. JNAME must match the N = parameter specified by the RJPWS initialization statement.

  TYPE = RJP
  : Specifies this as an RJP console.

  LL = nnn
  : Specifies the longest line length to be printed on the console. If the actual line length exceeds this length, the line is broken as a comma or a blank and continued on the next line. Specify **80**.

- RJPWS — Specifies the characteristics of one remote workstation.

  N = name
  : Specifies the name of the workstation.

  P = pswd
  : Specifies the password of this workstation.

  RD = nn
  : Specifies the number of readers at the workstation. Specify **1**.

  PR = nn
  : Specifies the number of printers at this workstation. Specify **1**.

  PU = nn
  : Specifies the number of punches at this workstation. Specify **1**.

  C = S|R
  : Indicate that the console and printer are separate devices. Specify **R**.

  COMPACT = compaction table name
  : Specifies the name of the default compaction table to be used for this workstation. **Do not specify**.

  AUTO = Y|N
  : Specifies whether JES3 is to initiate the connection. Specify **N**.

- DEVICE — Specifies a device other than the default. You must include a DEVICE statement for each device you want to change.

  DTYPE = RMTPRINT|RMTPUNCH
  : Choose one, depending on whether you use a printer or a punch data set.

  JNAME = name
  : Use the name you coded in the RJPWS statement as the work station name (N = parameter of the RJPWS statement, followed by the PR1 for printer or PU1 for punch.

CKPNT=nnnn
: Describes where there is a checkpoint after specifying the number of records. You can choose the number you want.

FORMS=YES,forms
: Specifies the name of the first forms on your remote printer. If you type STANDARD or default this parameter, you receive the standard forms defined on the OUTSERV initialization statement.

HEADER=NO/YES
: Describes whether or not you want header for a job.

CARRIAGE=YES,carrtape
: Describes the name of the first FCB associated with your remote printer.

COMPACT=comtab
: Describes the name of the compaction table in the COMPACT statement defined in JES3. Leave blank unless you code a decompaction routine.

JES3 optionally sets certain variable BIND parameters based on the setting of the corresponding parameters in the mode table entry specified for an LU via the VTAM MODETAB definitions. The following parameters on the MODEENT macro are significant to JES3:

FMPROF=X'03'
: This required parameter specifies the function management profile supported by SNA RJP to the workstation.

TSPROF=X'03'
: This required parameter specifies the transmission management profile supported by SNA RJP to the workstation.

PRIPROT=X'Aa'
: This parameter specifies the application program-to-LU protocol.

  A — Required subparameter that indicates multiple request unit chains. If you specify any value other than 'A', JES3 ignores it and forces the value to 'A'.

  a — This subparameter must be 1 or 3. If 1, it specifies primary transmission of end bracket. Specifying 3 indicates primary transmission of end bracket and compression of outbound data (JES3 to job entry station). Specify **3**.

SECPROT=X'Aa'
: This parameter represents the secondary network addressable unit (NAU) (logical unit-to-application) protocol.

  A — Indicates multiple request unit chains. Required if different JES3 changes to X'A'.

  a — Must be 1 or 3. If 1, indicates secondary transmission of end bracket. If 3, indicates secondary transmission of end bracket and compression of inbound data (job-entry station to JES3). Specify **1**.

COMPROT=X'7b80'
: Represents protocols for sessions regardless of session direction. The 7 indicates function management headers, brackets, and conditional bracket termination are used. Specifying 1 for "b" causes use of alternate code; specifying 0 for "b" causes nonuse of alternate codes (specify **0**). The 80 indicates that you want half-duplex, flip-flop transmission with secondary winning contentions.

RUSIZE=X'8585'
: Specifies a maximum request unit size of 256.

PSERVIC=X'cc1ed000B100C000000100f0'
: Specifies device characteristics:

cc	Can be 00 or 01 and specifies the LU profile 1.
e=0	Indicates that basic control and cards cannot span request units.
e=1	Indicates that basic control and cards can span request units. Set to **1**.
d	If equal to 1, indicates the primary will not compact or send PDIR. If equal to 2, indicates the primary will not compact but will send PDIR. If equal to 4, indicates the primary will compact but will not send PDIR. If equal to 6, indicates the primary will compact and send PDIR. Set to **2**.
f	If equal to 0, indicates that card media is not supported inbound. If set to 4, indicates that card media is supported inbound. Set to **4**.

Defining $RJESNA for OS/VS1 (RES)

This section contains reference information on defining $RJESNA workstations to remote entry service (RES). Two macroinstructions, TERMINAL and RTAM, contain parameters that define the $RJESNA workstations. (You do not need a LINE macroinstruction if all the workstations are SNA/SDLC.)

TERMINAL Macro

The TERMINAL macro instruction associates an identification number with a workstation and specifies the terminal device characteristics of the workstation. Specify at least one TERMINAL macro for each workstation. The macro parameters are as follows:

TDESCR=(w,t,d,f)
: Where w=printer size (3 for 132 character width), t=8 for SNA terminal, d=5 indicating SNA character string support, and f=3 indicating line printer and console function as well as punch transparency. Specify **TDESCR=(3,8,5,3)**.

RDRS=1
: Indicates support for the RJE reader function.

PTRS=1
: Indicates the number of printers at this workstation.

PCHS=1
: Indicates support of the RJE punch function.

PLGN = 0
: Indicates permanent logon is not used.

BUFXSIZE = 256
: Indicates the size of the $RJESNA buffer.

VBUF = tpnum
: Indicates the number of buffers to be used for input and output operations between the host and the $RJESNA workstation. Two buffers are always allocated for input. The number of output buffers establish the chain size. The value should be set between **6** and **14**.

NODE = name
: Specifies the symbolic name of this workstation as known to VTAM. These names must match the names specified on the VTAM LU definitions.

SESSLIM = value
: Specifies the maximum number of LU-to-LU sessions that can be active. This number should correspond to the number of RJE LUs available.

CPACTBL = name|NO
: Indicates the default compaction table. Specify **NO**.

RTAM Macro

The RTAM macro instruction specifies system-wide values, such as the maximum number of concurrently active readers and printers. Parameters that are important to consider in defining an RJE environment to RES are:

SNACOMP = YES
: Specifies use of compression for data transmitted to the workstations.

CPACT = NO
: Specifies nonuse of compaction.

CPACTDF = NO
: Specifies nonuse of compaction table.

APPLID = name
: Specifies the name the host assigned to RTAM, which corresponds to the VTAM APPL definition.

BIND Image

The BIND image for RJE, as defined by assembling a series of MODETAB, MODENT, and MODEND macros and link editing the resultant output into the SYS1.VTAMLIB data set, requires the following subparameters:

LOGMODE=name
: Where *name* corresponds to the mode table entry name BATCHS1. You can also modify the name of the mode table entry name by changing $RJESNA, as described in "VTAM Definitions" on page 17-6.

FMPROF=X'03'
: Required.

TSPROF=X'03'
: Required.

PRIPROT=X'A3'
: Specifies multiple request unit chains (required), definite response chains (required), compression (optional), and primary transmission of end bracket.

SECPROT=X'A1'
: Specifies multiple request unit chains (recommended), definite response chains (required), no compression (required), and secondary transmission of end bracket (required).

COMPROT=X'7080'
: Specifies function management headers (required), brackets (required), bracket termination rules (required), ASCII not used (required), and half-duplex flip-flop transmission (required).

RUSIZE=X'8585'
: Specifies a maximum request unit size of 256 bytes. RTAM assumes these values if specified as zero.

PSERVIC=X'01102000F100C08000010040'
: Specifies LU profile 1 (required), function management headers (required), no compaction but peripheral data records will be transmitted (optional), use of formatting characters (required), document output (optional), use of transparent record separator characters (default), half-duplex flip-flop protocol (recommended), and card format (optional).

  Use of optional features is not implied by their specification in the PSERVIC operand. If, for example, you set the optional card format on, and start the AUTORDR feature, all further console traffic from the host may cease. (This is because the reader tries to use LU1, but RES sends messages only on LU1. Prohibiting card format on LU1 resolves this problem.) If you set all bits in the PSERVIC field to zero, RTAM assumes the value X'01100000F100C08000010040', which specifies LU profile 1, function management headers, no compaction, no peripheral data information records, use of formatting characters, document output, use of transparent and record separator characters, and card format.

Defining $RJESNA for DOS/VSE (VSE/POWER)

The definition of the DOS system control program should include, on the SUPVR macro, the specification of POWER, multitasking support, and the VTAM access method. You accomplish the rest of the definition of the system to include support for $RJESNA with two macros: POWER and PRMT.

POWER Macro Parameters

The parameters to specify on the POWER macro are:

SNA = (lucount,password,applid)
: Where *lucount* allows control of VSE/POWER storage requirements (optional) and specifies the approximate number of concurrent SNA sessions, *password* represents the optional POWER application password, and *applid* defines the POWER application name as defined to VTAM.

PRMT Macro Parameters

The parameters to specify on the PRMT macro are:

REMOTE = nnn
: Where *nnn* specifies the remote identifier. The first SNA remote identifier must be greater than the highest BSC remote identifier.

TYPE = LUT1
: Indicates SNA support.

REF = nnn
: Where *nnn* specifies the remote identifier of another PRMT macro that defines the workstation characteristics.

CONSOLE = YES
: Indicates the presence of a console function at the remote workstation.

PSWRD = password
: Indicates the password that may accompany a logon request from the workstation. If specified, the password entered by the remote operator must match this password.

CMPACT = name
: Specifies the default compaction table name. Should not specify.

LU = name
: Specifies the secondary LU name that uses this remote identifier. Should correspond to the names defined to VTAM. If you do not specify this operand, LU name checking is not performed.

BIND Image

You define the BIND image for RJE by assembling a series of MODETAB, MODEENT, and MODEEND macros and link editing the resulting output into the core-image library.

The following MODEENT parameters are significant in defining an RJE environment to DOS/VSE:

LOGMODE=name
: Where *name* corresponds to the mode name specified by the $RJESNA operator at sign on time (in response to the ENTER HOST ID message.)

FMPROF=X'03'
: Required.

TSPROF=X'03'
: Required.

PRIPROT=X'A3'
: Specifies multiple request unit chains (required), definite response chains (required), compression (optional), and primary transmission of end bracket.

SECPROT=X'A1'
: Specifies multiple request unit chains (recommended), definite response chains (required), no compression (required), and secondary transmission of end bracket (required).

COMPROT=X'7080'
: Specifies function management headers (required), brackets (required), bracket termination rules (required), ASCII not used (required), and half-duplex flip-flop transmission (required).

RUSIZE=X'8585'
: Specifies a maximum request unit size of 256 bytes. POWER enforces these values.

PSERVIC=X'01102000F100C00000010040'
: Specifies LU profile 1 (required), function management headers (required), no compaction but peripheral data records transmitted (optional), use of formatting characters (required), document output (optional), use of transparent record separator characters (default), and card format (optional).

  Use of optional features is not implied by their specification in the PSERVIC operand. If you set all bits in the PSERVIC field to zero, POWER assumes the value X'01100000F100000000010040', which specifies LU profile one, function management headers, no compaction, no peripheral data information records, use of formatting characters, document output, use of transparent and record separator characters, and card format.

Chapter 18. X.21 Circuit Switched Network Support

The purpose of this chapter is to:

- Introduce you to X.21 circuit switched network support (X.21).
- Explain how you can make the connection to X.21 circuit switched network support when using the Series/1 EDX SNA.

Note: Throughout the rest of this chapter, we refer to X.21 circuit switched network support as X.21.

> **Warning:** The information in this chapter is only applicable when using the SDLC support available in the SNA program product.

What Is X.21?

X.21 is an extension to EDX SNA; it allows you to link from a Series/1 into a **digital public data network**.

X.21 is the international standard designated by many countries in Europe and the Far East as the basis for their digital communications networks.

You must know the following information before using X.21 circuit switched network support:

- The network your country uses
- The prerequisites for the network requirements.

How Does X.21 Work?

Series/1 X.21 circuit switched network support involves the following series of operations:

- Establishing outbound calls with retries and delays that you specify.
- Accepting incoming calls.
- Monitoring network status through call progress signals and network provided information.
- Terminating a connection.

How Do I Make a Connection?

In order to establish a connection using X.21, you must perform the following steps:

1. Meet the X.21 hardware and software requirements.
2. Choose a connection type.
3. Define your SNA network.
4. Define your connection record within the X.21 data set.
5. Start the system error log.

How Do I Terminate a Connection?

You terminate an X.21 connection by deactivating the EDX SNA network.

X.21 Software Requirements

In order to make or accept a call, X.21 requires that you have a reserved data set named $$X21DS (for X.21 use only) on your IPL volume (EDX002). This IBM-supplied one-record data set contains no information until you create connection records. This data set stores all connection records used by X.21. See "Creating a Connection Record" on page 18-4 for information on creating a connection record. In order to edit, create, and format $$X21DS, you must use the $FSEDIT utility. Refer to the *Operator Commands and Utilities Reference* for instructions on how to use the text editor. See "Creating a Connection Record" on page 18-4 for information on X.21 connection records.

X.21 Hardware Requirements

You must also make sure that the IBM 2080 Synchronous Communication Single-Line Control High Speed Feature Card is included with your Series/1 hardware package. This card must be attached and jumpered before implementing X.21. Refer to the *IBM Implementation of X.21 Interface General Information Manual* for information on connecting data terminal equipment (DTE) to data communication equipment (DCE). Refer to the *IBM Series/1 Communication Features Theory Diagrams* for information on jumpering this card.

Defining Your SNA Network

X.21 is an extension of EDX SNA. Before you can define X.21, you must define SNA.

You define the SNA network differently when X.21 is part of the system. Two operands are affected when defining the SNAPU configuration statement.

- The CNCTYPE operand
- The CNCNAME operand.

Before you can define the SNA network, you must choose a connection type and create a name for the connection record. After choosing the type and name, use the CNCTYPE and CNCNAME operands to define the SNAPU statement.

Choosing a Connection Type

In order to establish connection, you must create a physical communications path. This path paves the way for subsequent message and command exchange during a session. EDX SNA provides six connection types (CNCTYPEs) to create the path for a switched network. See "SNAPU Configuration Statement" on page 2-5 for an explanation of these types.

When using X.21, you must choose between three connection types. In the case of an outbound connection, the Series/1 issues an *auto call* or *direct call* to create the path. To accept an incoming call, the Series/1 issues an *auto answer* to create the path.

If you define CNCTYPE as one of the six used in EDX SNA, SDLC converts the type to an X.21 connection method. The following table shows how SDLC maps the connection types for operation with X.21.

SDLC CONNECTION TYPE (supplied on SNAPU)	X.21 CONNECTION METHOD	SDLC PROTOCOL AFTER CONNECTION
Auto answer (AA)	Auto answer (AA)	Pt-to-pt
Auto call (AC)	Auto call (AC)	Pt-to-pt
Direct call (DC)	Direct call (DC)	Pt-to-pt
Manual call (MC)	Auto call (AC)	Pt-to-pt
Manual answer (MA)	Auto answer (AA)	Pt-to-pt

Description of Auto Call and Direct Call

If you select auto call as your CNCTYPE, and the 2080 attachment card is jumpered for switched operation, X.21 obtains the connection record from the X.21 data set ($$X21DS). You must build a connection record in the X.21 data set for any auto call you attempt. The name of the connection record must be the same as the name you choose for the CNCNAME parameter on SNAPU statement. If you leave the CNCNAME parameter blank, the connection record name must be X21RECyy.

Note: *yy* is the device address of the 2080 attachment card in hexadecimal.

When you specify direct call as your CNCTYPE and the 2080 attachment card is jumpered for switched operation, X.21 makes a connection directly to a predefined number. If you code a name for the CNCNAME parameter, you must code the same name for the connection record. Direct calls do not require an address on a per call basis, so you need only to code the retry and delay fields. You must specify the connection record name on the SNAPU statement by defining the CNCNAME operand. If you leave CNCNAME blank, X.21 does not search for the default connection record (X21RECyy). When attempting connection, X.21 assumes the defaults for the retry and delay fields. (They are 1 and 0, respectively.) See Chapter 2, "Installing and Defining the SNA Network for Use with SDLC" for a full description of how to define the network.

Figure 18-1 is a partial coding example of the SNAPU statement. It includes the CNCTYPE and CNCNAME operands.

There should be only *one* complete SNAPU definition (see "SNAPU Configuration Statement" on page 2-5).

```
NET1     CSECT ,                  DEFINE THE NETWORK
         SNAPU
               CNCTYPE=AC,        CONNECTION TYPE                    C
               CNCNAME=CONREC,    CONNECTION RECORD FOR X.21         C
                    •
                    •
                    •
```

Figure 18-1. SNAPU Statement Coding Illustration

Creating a Connection Record

You can create an unlimited amount of connection records in the X.21 connection data set ($$X21DS). Each record must begin on a separate line in the data set. In order to define your connection record you must follow the format depicted below:

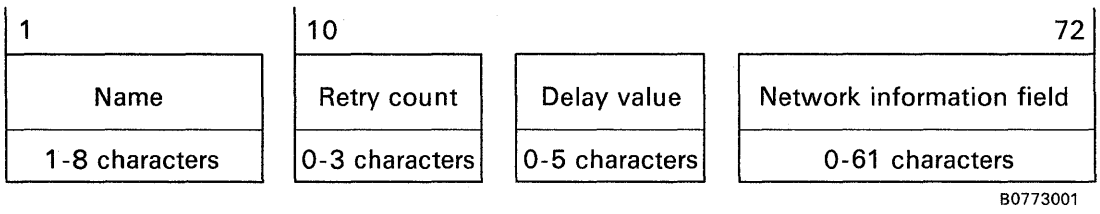


Figure 18-2. Connection Record Format

Name

The 1 – 8 alphanumeric character name of the connection record that the X.21 network uses when making a call. If you define the name with fewer than 8 characters, you must pad with blanks so that you fill columns 1 through 8. If you choose to use the default record with auto call, one of your record names must be X21RECyy (where *yy* is the hexadecimal address of the 2080 card).

Retry count

The maximum number of retries allowed for either an auto or direct call after a call progress signal occurs. This field must be a decimal number ranging from 0 to 255. To use the default, which is 1, you must code a comma.

Note: This field must start in column 10.

Use a comma to separate the retry count from the delay value.

Delay value

The time it takes in milliseconds for X.21 to reissue an auto or direct call after receiving an error from a call progress signal. This field must be a decimal number ranging from 0 to 65535. To use the default, which is 0, you must code a comma. Use a comma to separate the delay value from the network information field.

Network information field

Facility requests or address selections. Modify this field on the basis of the network you select. The maximum length of this field is 61 characters. To use the maximum length of 61 characters, you must code the default for retry and delay values. This field is not required on a direct call. You must code a comma between the delay value and network information field even if you leave the network information field blank.

Note: The information used in this field is network dependent. Refer to the *IBM Implementation of X.21 Interface Manual* for a description of IBM's implementation of CCITT recommendation X.21.

See Figure 18-3 for an illustration of an $FSEDIT screen containing five connection records.

```
                          •
                          •
                          •
00080 CONNREC1 255,65000,NETWORK INFORMATION FIELD+
00090 CONNREC2 255,100,
00100 CONNREC3 ,,NETWORK INFORMATION FIELD+
00110 X21RECOA 1,2,NETWORK INFORMATION FIELD+
00120 RECORD   25,255,NETWORK INFORMATION FIELD+
```

Figure 18-3. Coding Illustration of Connection Records

Note: CONNREC2 is an example of a direct call.

X.21 Error and Call Progress Signal Logging

You should ensure that $LOG is active when using X.21 circuit switched network support. Then use the LL or LP command of the $DISKUT2 utility to list your error log record. The system tells you when you have an X.21 switched error. Then you need to check the error log record to determine what the error is.

Figure 18-4 shows an example of the printed output created by the $DISKUT2 utility when you have X.21 circuit switched network support. In this case, the X.21 return code field (word 36) equals −9 (FFF7), indicating a read instruction error. An explanation of the numbered items follows the example.

```
    COMMAND(?): LL
1   LOG DS NAME:  EDXLOGDS

    DEVICE ADDRESS (NULL FOR ALL):  000A
2   I/O LOG ERROR COUNTERS (BY DEVICE ADDR):
                    •
                    •
                    •
    REQUESTED HEX DUMP OF LOG RECORD:
              3
    0000      0100 0000 000A 7E00 0000 0222 0000 0000
                                              4  5  6
    0010      0000 0000 0000 0000 0002 02D0 040A 8000
              7  8  9
    0020      FFFF 0005 FFF7 0000 0000 0000 0000 0000
    0030      0000 0000 0000 0000 0000 0000 0000 0000
    0040      0000 0000 0000 0000 0000 0000 0000 0000
    0050      0000 0000 0000 0000 0000 0000 0000 0000
    0060      0000 0000 0000 0000 0000 0000 0000 0000
    0070      0000 0000 0000 0000 0000 0000 0000 0000
    0080      0000 0000 0000 0000 0000 0000 0000 0000
    0090      0000 0000 0000 0000 0000 0000 0000 0000
    00A0      0000 0000 0000 0000 0000 0000 0000 0000
    00B0      0000 0000 0000 0000 0000 0000 0000 0000
    00C0      0000 0000 0000 0000 0000 0000 0000 0000
    00D0      0000 0000 0000 0000 0000 0000 0000 0000
    00E0      0000 0000 0000 0000 0000 0000 0000 0000
    00F0      0000 0000 0000 0000 0000 0000 0000 0000

    LOG LISTING ENDED
```

Figure 18-4. Example of the X.21 Printed Log Information for a Read Error

1 This is the name of the log data set specified with $LOG. In this example, the log data set is EDXLOGDS.

2 This is what you see on the usual log information. The bullets replace the list of device addresses and I/O error indications that $DISKUT2 provides. These device addresses range from X'00' to X'FF' or from 0 to 255.

3 This is the start of your X.21 log record output. The first 28 bytes contain the common log area for a recoverable SDLC device error.

4 This byte contains the X.21 log record type, X'04'. It marks the beginning of the X.21 statistical log.

5 This byte contains your device address, in this case X'0A'.

6 This word contains the X.21 error flags reserved for system use. In this case, it indicates that there are X.21 log entries.

7 This word indicates that there is a read instruction error when equivalent to −1. (X'FFFF'). Refer to this byte only when the word indicated by **9** equals −9 decimal (X'FFF7').

8 This word contains the error return code from the READ instruction. Refer to this word only when the word indicated by **9** equals −9 decimal (X'FFF7'). (Refer to *Messages and Codes* for the definitions of the Disk READ/WRITE error codes.)

9 When this word equals −9 (X'FFF7'), you must consult the two words indicated by **7** and **8**. In this case, the remainder of the log record contains zeroes.

Figure 18-5 shows a second example of the printed output created by $DISKUT2 when you have X.21 circuit switched network support. In this case, the X.21 return code field (word 36) equals −27 (X'FFE5') and indicates a device error. An explanation of the numbered items follows the example.

```
    COMMAND(?): LL
1   LOG DS NAME:  EDXLOGDS

    DEVICE ADDRESS (NULL FOR ALL):  000A
2   I/O LOG ERROR COUNTERS (BY DEVICE ADDR):
                    •
                    •
                    •
    REQUESTED HEX DUMP OF LOG RECORD:
              3
    0000      0100 0000 000A 7E00 0000 0222 0000 0000
                                           4  5  6
    0010      0000 0000 0000 0000 0002 02D0 040A C000
                        7  8  9
    0020      0000 0000 FFE5 03FA 0000 0000 0000 0000
                                             10
    0030      0000 0000 0000 0000 0000 0000 0002 0000
    0040      0000 0000 0000 0000 0000 0000 0000 0000
    0050      0000 0000 0000 0000 0000 0000 0000 0000
                        11
    0060      0000 0000 0001 0000 0000 0000 0000 0000
    0070      0000 0000 0000 0000 0000 0000 0000 0000
                                        12
    0080      0000 0000 0000 0000 0000 0000 0000 0000
    0090      0000 0000 0000 0000 0000 0000 0000 0000
    00A0      0000 0000 0000 0000 0000 0000 0000 0000
    00B0      0000 0000 0000 0000 0000 0000 0000 0000
    00C0      0000 0000 0000 0000 0000 0000 0000 0000
    00D0      0000 0000 0000 0000 0000 0000 0000 0000
    00E0      0000 0000 0000 0000 0000 0000 0000 0000
    00F0      0000 0000 0000 0000 0000 0000 0000 0000

    LOG LISTING ENDED
```

Figure 18-5. Example of the X.21 Printed Log Information for a Device Error

1 This is the name of the log data set specified with $LOG. In this example, the log data set is EDXLOGDS.

2 This is what you see on the usual log information. The dots (•) replace the list of device addresses and I/O error indications that $DISKUT2 provides. These device addresses range from X'00' to X'FF', or 0 to 255.

3 This is the start of your X.21 log record output. The first 28 bytes contain the common log area for a recoverable SDLC device error.

4 This byte contains the X.21 log record type, X'04'. It marks the beginning of the X.21 statistical log.

5 This byte contains your device address, in this case X'0A'.

6 This word contains the X.21 error flags reserved for system use. In this case, it indicates that there are X.21 log entries and that a device error occurred.

7 This word is the X.21 return code field. When it equals −27 (X'FFE5'), consult the device error code field indicated by **9**.

8 This byte shows you how many times the call was retried before it failed. In this case, the retry count equals 3.

9 This byte gives you the device error code, in this case −6 (X'FA'). Use the data in this byte only when the word indicated by **7** equals −27. The device error codes are as follows:

−1 (X'FF') Buffer overrun

−2 (X'FE') Unsuccessful DCE clear

−3 (X'FD') Interface data check error

−4 (X'FC') Invalid interrupt code

−5 (X'FB') Invalid interrupt status byte

−6 (X'FA') Invalid I/O condition code

−7 (X'F9') Start cycle steal status issued

−8 (X'F8') Specification check.

Note: Refer to the hardware manual *IBM Series/1 Communications Theory Diagrams* for the meanings of these messages.

The next 100 bytes (00 to 99) are the call progress signal counters. They record call progress signals and hardware errors. The following example shows the meaning of the significant bytes. The number within the byte indicates how many times the error occurred. For example, **10** shows you a call progress signal of 21 because it's the twenty-first byte after **9** (X'28' - X'3D'); the number 02 within the byte tells you that it occurred twice. **11** shows you a call progress signal 61 because it's the sixty-first byte after **9** (X'28' - X'65'); the number 01 within the byte tells you that it occurred once. The byte number column gives the byte number relative to the beginning of the log indicated by **3**.

Call progress signal	Byte number	Meaning of signal	What X.21 does
00 01 02 03	28 29 2A 2B	Reserved Terminal called Redirected call Connect when free	Does not clear. Waits for attempt to complete.
20 21 22 23	3C 3D 3E 3F	No connection Number busy Selection signals procedure error Selection signal transmission error	Clears due to short-term conditions. Tries again up to retry limit.
41 42 43 44 45 46 47 48 49 51 52	51 52 53 54 55 56 57 58 59 5B 5C	Access barred Changed number Not obtainable Out of order Controlled not-ready Uncontrolled not-ready DCE power off Invalid facility request Network fault in local loop Call information service Incompatible user class of service	Clears due to long-term conditions. Call unsuccessfully completed.
61	65	Network congestion	Clears due to network short-term conditions. Tries again up to retry limit.
71 72	6F 70	Long-term network congestion RPOA out of order	Clears due to long-term network conditions. Call unsuccessfully completed.
81 82 83	79 7A 7B	Registration/cancellation confirmed Redirection activated Redirection deactivated	Clears due to DTE network procedure.

Figure 18-6. Call Progress Signal Counter Usage

12 This byte marks the end of the statistical log.

X.21 Return Codes and Messages

Following are the return codes you may receive from the X.21 circuit switched network support. See Appendix A, "Messages" for an explanation of any messages you receive from X.21.

Return Code	Condition
-32	System is unable to find X.21 support. IPL the system.

-31	Not enough storage available to handle the number of X.21 requests. Use the $DISKUT2 SS command to allocate more storage. You can issue three simultaneous requests for every 256 bytes of storage allocated.
-30	Your supervisor does not contain X.21 support.
-29	System does not have enough storage available to load the X.21 support or the connection record data set, $$X21DS is not on the IPL volume (EDX002).
-27	Unrecoverable hardware error. If $LOG is active, check the error log record for the X.21 device for more details.
-26	Hardware error for the 2080 feature card.
-25	Connection failed.
-24	Time expired for the completion of a call request. Call request failed.
-23	You cancelled a call request with an SNADACT or $C command.
-22	Call request failed due to Public Data Network problems. Call progress signals were invalid.
-21	Call request failed due to Public Data Network problems. Call progress signals were incomplete.
-20	Call request failed and network would not allow request to be retried. If $LOG is active, check the error log record for the X.21 device for more details.
-19	Number of retries exhausted for the call request. If $LOG is active, check the error log record for the X.21 device for more details.
-18	Hardware error for the 2080 feature card. I/O request could not be completed.
-17	No call request in progress.
-16	Network information field has no plus sign or just a plus sign.
-15	The value in the retry or delay field exceeds the maximum value allowed.
-14	The retry or delay field contains a negative value.
-13	A comma must separate the retry, delay, and network information fields.
-12	The retry or delay field contains an invalid character.
-11	System does not have enough storage to execute a call request.
-10	Not enough storage in partition 1 for X.21 to execute a request.
-9	An EDL instruction failed. If $LOG is active, check the error log record for the X.21 device to find the failing instruction.
-1	Successful completion.
1	Registration or cancellation request processed.
2	Redirection activated.
3	Redirection deactivated.

Storage Requirements for SNA with SDLC and X.21

Figure 18-7 describes the storage requirements for EDX SNA with SDLC and X.21 within a Series/1. It describes the calculations needed to estimate the storage requirements of your customized configuration. With Version 2.1 of EDX SNA, the PU control blocks for each PU can be in separate partitions, and $SNA need not be in the same partition of any PU. When $SNA is loaded, SNA allocates a 1-page program control data area with a label of **DATA** in partition number 1.

Description	# of bytes
Storage requirements for $SNA: SNA resident code with X.21 support	 39,282
Storage requirements for each PU ($NETx): Resident code to support the PU PU control blocks Stack area # of stacks (STKNUM) x 348 bytes DCB area # of DCBs (DCBNO) x 18 bytes LU area (# of LUs (SNALU x 350 bytes) + (8 bytes x # of CTEs (CTE))) Buffer area (# of buffers (BUFNO + SENDBUF + RECVBUF) x buffer size (BUFSIZ + 24)) Total storage for PU (round up to page boundary)	 4072 884
Storage requirements for each SNA application: $NETCMD $NETPACT	 2664 1190

Figure 18-7. Storage Requirements

Sample Storage Calculation

Storage requirements for $SNA:	
SNA resident code with X.21 support	38.5K bytes
Storage requirements for each PU ($NETx):	
Resident code to support the PU	4072 bytes
PU control blocks	884 bytes
Stack area (2 x 348 bytes)	696 bytes
DCB area (2 x 18 bytes)	36 bytes
LU area (1 x (350 bytes + (8 bytes x 2)))	366 bytes
Buffer area ((3 + 2 + 2) x (256 + 24 bytes))	1960 bytes
Total storage for PU (rounded to page boundary)	8.25K bytes
Storage requirements for each SNA application	
$NETCMD	2664 bytes
$NETPACT	1190 bytes

Figure 18-8. Sample Storage Calculation

Appendix A. Messages

Introduction to Messages

Some messages explained in this appendix indicate abnormal or error conditions. Others provide information during normal operations. Most messages require you to take some action, either to restore conditions to normal or to allow normal operations to proceed.

Finding Messages in this Appendix

All messages in this appendix are grouped by message identifiers. These identifiers begin with a "$" followed by other characters (for instance, NET or SNA). Within these groups, the messages are listed in alphabetical order.

An example of finding a message follows. Assume that the following message appears on your terminal:

```
$NETx NETWORK ACTIVATION IN PROGRESS
```

You find the group of messages with the **$NETx** identifier. Then you look at the message itself. The first letter in the message is **N**. You then look for the message under the listings for **N** (alphabetical order). You find the message documented the same way it was displayed.

Any messages that begin with numbers or special characters appear at the end of the group.

Some messages in this appendix contain variables. Variables are names that can change within the text of the message such as return code, volume, device type, or data set.

Variables are represented in this book as follows:

xxx = First or only variable in a message

yyy = Second variable in a message.

Types of Messages Documented in this Appendix

All messages explained in this appendix are issued from $SNA and are those you receive on the operator's console while the SNA application is running. Most of these messages are for your information only. Some messages do require that you correct a problem that SNA encountered in the network.

SDLC messages that you receive when using the shared SDLC support are explained in *Messages and Codes.*

$RJESNA messages, those you receive when using the $RJESNA remote job utility, are explained in the *Messages and Codes.*

$EDXASM messages, those you might receive during either SNA network generation or when assembling your SNA application, are also explained in the *Messages and Codes.*

How Messages are Documented

The documentation for each message includes the following:

Message Text: The "message text" appears in this book in boldface type, as it is actually printed or displayed. (Lowercase variable fields are replaced with correct data when the message appears.)

Return Codes: The "message text" sometimes includes a return code, which is generated by the EDX. The name of the function that generated the return code is specified in the "issued by" portion of the message documentation.

Issued By: The "issued by" text identifies the function that issued the message.

Explanation: The "explanation" identifies all the variable fields and describes any information supplied by the program. The conditions under which the message appears are noted.

System Action: The "system action" describes how the system reacts to the condition that caused the message to be issued.

Response: The "response" explains how you are to reply to the message or what, if any, corrective actions you must take to help you resume operation.

$NETx BUFNO MUST BE AT LEAST ONE GREATER THAN THRESH

Issued By: $NETx

Explanation: You must specify a value for BUFNO on the SNAPU statement that is at least one greater than the value specified for THRESH for PU *#x*.

System Action: The $NETx program unloads.

Programmer Response: Recode the SNAPU statement to specify BUFNO greater than the value for THRESH and regenerate the PU using BLDNETx.

$NETx CLOSE REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process a CLOSE request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services error or the Data Link Router error. For any other return code, refer to *Messages and Codes* manual for a description of the shared SDLC error.

$NETx CONTROL CONTACT REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process a CONTROL CONTACT request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services error or the Data Link Router error. For any other return code, refer to *Messages and Codes* manual for a description of the shared SDLC error.

$NETx CONTROL TEST REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process a CONTROL TEST request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services error or the Data Link Router error. For any other return code, refer to the *Messages and Codes* manual for a description of the shared SDLC error.

$NETx CONTROL XID REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process a CONTROL XID request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services error or the Data Link Router error. For any other return code, refer to the *Messages and Codes* manual for a description of the shared SDLC error.

$NETx LINK LEVEL ACTIVATED VIA SNA

Issued By: $NETx

Explanation: This message appears after Series/1 SNA support for the PU has been loaded and the resources required to operate the PU have been obtained. The Series/1 SDLC station for this PU has been set to normal response mode by the primary SDLC station.

System Action: Normal processing continues.

Response: None.

$NETx LOAD OF DLC PROGRAM FAILED FOR DEVICE name.
EDL LOAD RC = xxx

Issued By: $NETx

Explanation: An error was detected because the EDL LOAD instruction failed for the shared SDLC support.

System Action: The $NETx program unloads.

Programmer Response: Refer to the *Messages and Codes* manual for a description of the load error. Correct the error, and reactivate the PU using the PUACT attention list command.

$NET*x* NETWORK ACTIVATION IN PROGRESS

Issued By: $NETx

Explanation: SNA support has begun activating PU #*x*.

System Action: SNA support proceeds with PU activation.

Response: None.

$NET*x* NETWORK DEACTIVATED

Issued By: $NETx

Explanation: SNA support has finished deactivating PU #*x*

System Action: SNA support either unloads the PU or attempts to reactivate the PU depending on how you defined the UNLOAD operand on the SNAPU statement.

Response: None.

$NETx NETWORK DEACTIVATION ALREADY IN PROGRESS

Issued By: $NETx

Explanation: You tried to deactivate PU #*x* from the Series/1 by using the SNADACT or PUDACT operator command. The system rejects this command because the PU is already in the process of deactivating.

System Action: SNA support continues to deactivate the PU.

Response: None.

$NETx NETWORK DEACTIVATION IGNORED – SESSIONS ACTIVE

Issued By: $NETx

Explanation: SNA support has received a request to deactivate PU #*x* normally, but there are still active sessions in progress for that PU.

System Action: SNA support ignores the request to deactivate the PU.

Response: In this situation you have several choices:

- You can deactivate the PU from the host, or stop all sessions and have SNA support attempt to deactivate normally again.
- You can also use the SNADACT or PUDACT operator command to deactivate the PU abnormally.

If you decide you do not wish to deactivate the PU, ignore this message.

$NETx NETWORK DEACTIVATION STARTED

Issued By: $NETx

Explanation: A PUDACT or SNADACT command was entered, and the deactivation process for PU #*x* has been initiated.

System Action: The resources that are reserved for this PU are deallocated. No new users are allowed to open sessions for this PU.

Response: None.

$NETx NO BUFFER AVAILABLE TO SEND REQDISCONT

Issued By: $NETx

Explanation: SNA support is attempting to deactivate PU #*x*. The PU cannot be deactivated because there is no SNA buffer to send REQDISCONT to the host.

System Action: SNA support waits until a buffer is available. This buffer is then used to send REQDISCONT.

Response: You can either deactivate the PU from the host or wait for SNA support to free a buffer allowing SNA to send REQDISCONT.

$NETx OPEN REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process an OPEN request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services error or the Data Link Router error. For any other return code, refer to the *Messages and Codes* manual for a description of the shared SDLC error.

$NETx READ REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process a READ request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services Error or the Data Link Router error. For any other return code, refer to the *Messages and Codes* manual for a description of the shared SDLC error.

$NETx SESSIONS RESET − LINK LEVEL ACTIVE

Issued By: $NETx

Explanation: The host detected a serious error with PU #*x* and is reinitiating PU activation. The host restarts SDLC by sending a SDLC SNRM command.

System Action: SNA support clears the system of all sessions for the PU and resets all its control blocks for SDLC and SNA for the PU.

Response: You can restart lost sessions by reissuing NETINIT instructions. You can find out what error the host encountered by using the host network problem determination utilities.

$NETx SNA/SDLC CONFIGURATION ERROR.
RETURN CODE = xxx

Issued By: $NETx

Explanation: SNA support detected an error for PU #*x*. The return code indicates the type of error.

System Action: SNA support ends all sessions for the PU and either unloads or attempts to reactivate the PU, depending on how you previously defined the UNLOAD parameter on the SNAPU statement.

Response: Check the return code description for the cause of the error. See "SDLC Error Return Codes" on page 2-21 for a list of return codes with their explanations.

$NETx SNA/SDLC LINE IS A MANUAL CALL
PLEASE DIAL YOUR NETWORK EXTENSION

Issued By: $NETx

Explanation: SNA support is trying to communicate over a switched line using PU #*x*. In order for SNA support to do this, you must first dial the appropriate network PU.

System Action: SNA waits for you to dial the network PU and establish connection. SNA support then proceeds with PU activation.

Response: If you have not dialed the network PU yet, do so. If you have dialed the network PU and connected successfully, disregard this message.

$NETx SNA/SDLC TERMINATED DUE TO A PERMANENT SDLC ERROR CONDITION
SDLC DEVICE RETURN CODE = xxx

Issued By: $NETx

Explanation: SNA support encountered an unrecoverable hardware error for PU #*x*. *xxx* indicates the type of error.

System Action: SNA support terminates all sessions for that PU and either unloads or attempts to reactivate the PU, depending on how you previously defined the UNLOAD operand on the SNAPU statement.

Response: Determine what the problem is and try to resolve it before reactivating the PU. See "SDLC Error Return Codes" on page 2-21 for more information on SDLC return codes.

$NETx SNA SYSTEM FAILURE – SNA TERMINATED DUE TO UNRECOVERABLE ERROR CONDITION

Issued By: $NETx

Explanation: $NETx either program checked or encountered an unrecoverable error condition.

System Action: The system deactivates the network and unloads $NETx. The Level Status Block (LSB) and Task Control Block (TCB) for $NETx are dumped.

Response: Please write an APAR.

$NETx SNA UNLOADING – NO SEND BUFFERS WERE SPECIFIED FOR BUFFER POOLING

Issued By: $NETx

Explanation: You must specify a value that determines the number of buffers (0 – 253) allowed for concurrent send operations for PU #x when using buffer pooling (BUFPOOL = YES) on the SNAPU statement. This value is set on the SBUFNO parameter on the SNAPU statement.

System Action: The $NETx program unloads.

Response: Recode the PU parameter statements to include send buffers and regenerate the PU using BLDNETx.

$NETx START PORT REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process a START PORT request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services error or the Data Link Router error. For any other return code, refer to the *Messages and Codes* manual for a description of the shared SDLC error.

$NETx STOP PORT REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process a STOP PORT request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services error or the Data Link Router error. For any other return code, refer to the *Messages and Codes* manual for a description of the shared SDLC error.

$NETx THIS PROGRAM IS INCOMPATIBLE WITH THE $SNA PROGRAM HAVING SDLC SUPPORT

Issued By: $NETx

Explanation: An error was detected because CNCTYPE=EX was coded on the SNAPU statement, and the $SNA program that was loaded contains SDLC support. CNCTYPE must not be coded as "EX" when using SDLC support.

System Action: The $NETx program unloads.

Programmer Response: See Chapter 2, "Installing and Defining the SNA Network for Use with SDLC" for an explanation of how to code the SNAPU statement to use the SDLC support in the SNA product.

$NETx THIS PROGRAM IS INCOMPATIBLE WITH THE $SNA PROGRAM HAVING EXTENDED DLM SUPPORT

Issued By: $NETx

Explanation: An error was detected because the CNCTYPE operand on the SNAPU statement was not coded as "EX," and the $SNA program that was loaded contains extended DLM support. CNCTYPE must be coded as "EX" when using extended DLM support.

System Action: The $NETx program unloads.

Programmer Response: See Chapter 3, "Installing and Defining the SNA Network for Use with Shared SDLC" for an explanation of how to code the SNAPU statement to use the shared SDLC and extended DLM support.

$NETx WRITE REQUEST FAILED FOR DEVICE name.
DLC RC = xxx

Issued By: $NETx

Explanation: An error was detected while attempting to process a WRITE request by the corresponding DLC support for the specified device.

System Action: The $NETx program unloads.

Programmer Response: If the return code is −2 through −49, refer to the *Messages and Codes* manual for a description of the Communication Common Services error or the Data Link Router error. For any other return code, refer to the *Messages and Codes* manual for a description of the shared SDLC error.

$NETx X.21 SW CONNECTION FAILED FOR DEVICE ADDRESS: *xxx*. **RETURN CODE =** *yyy*

Issued By: $NETx

Explanation: The X.21 Circuit Switched Network is unable to complete your call request for PU #*x*. (Note: *xxx* is the device address in decimal.) The type of error is indicated by *yyy*.

System Action: The $NETx program unloads.

Response: Check the X.21 Circuit Switched Network return codes for a description of the code returned in this message. Correct the problem and reactivate the SNA PU.

$NETx X.21 SW REDIRECTION ACTIVATED FOR DEVICE ADDRESS: *xxx*

Issued By: $NETx

Explanation: The X.21 Circuit Switched Network has processed your redirection activation request for PU #*x*. (Note: *xxx* is the device address in decimal.)

System Action: The $NETx program unloads.

Response: To continue processing, you can:

- Redefine the SNA PU, or
- Change the X.21 connection record for the SNA PU and reactivate the PU.

$NETx X.21 SW REDIRECTION DEACTIVATED FOR DEVICE ADDRESS: *xxx*

Issued By: $NETx

Explanation: The X.21 Circuit Switched Network has processed your redirection deactivation request for PU #*x*. (Note: *xxx* is the device address in decimal.)

System Action: The $NETx program unloads.

Response: To continue processing, you can:

- Redefine the SNA PU, or
- Change the X.21 connection record for the SNA PU and reactivate the PU.

$NETx X.21 SW REGISTRATION/CANCELLATION CONFIRMED FOR DEVICE ADDRESS: *xxx*

Issued By: $NETx

Explanation: The X.21 Circuit Switched Network has processed your registration or cancellation request for PU #*x*. (Note: *xxx* is the device address in decimal.)

System Action: The $NETx program unloads.

Response: To continue processing, you can:

- Redefine the SNA PU, or
- Change the X.21 connection record for the SNA PU and reactivate the PU.

$NETx X.21 SW TEST FAILED FOR DEVICE ADDRESS: *xxx*
RETURN CODE = *yyy*

Issued By: $NETx

Explanation: The 2080 feature card for PU #x is jumpered incorrectly or is malfunctioning. (Note: *xxx* is the device address in decimal.) The type of error is indicated by *yyy*.

System Action: The $NETx program unloads.

Response: Check the X.21 Circuit Switched Network return codes for a description of the code returned in this message. Correct the problem and reactivate PU #*x*. (Refer to the *IBM Series/1 Communication Features Theory Diagrams* for information on how to jumper the 2080 feature card for the X.21 Circuit Switched Network.)

$SNA DATA LINK ROUTER SUPPORT NOT INCLUDED IN SUPERVISOR.

Issued By: $SNA

Explanation: The $SNA program found that the Data Link Router module (DLCROUTE) was not included in the supervisor generated.

System Action: The $SNA program unloads.

Programmer Response: Regenerate the supervisor to include Data Link Router module (DLCROUTE).

$SNA ERROR READING SNA INITIALIZATION DATASET - SNAINIT
RETURN CODE = *xxx*

Issued By: $SNA

Explanation: An I/O error occurred when trying to read the SNAINIT data set on the IPL volume.

System Action: The system ignores the command if PUACT generated the message. If an SNA load command generated the message, the system stops the load process.

Response: Correct the problem indicated by the return code. Refer to the section on disk and diskette Read/Write return codes in *Messages and Codes* manual for an explanation.

$SNA FREEMAIN OF STORAGE IN PARTITION #1 FAILED

Issued By: $SNA

Explanation: The FREEMAIN instruction used to free the 256 bytes of storage obtained from partition 1 failed while $SNA was deactivating the network.

System Action: $SNA unloads.

Response: Perform an initial program load (IPL) of the operating system to recover the storage lost. If you can accept the loss of 256 bytes of storage from partition 1, you do not need to IPL again.

$SNA FULLMSG SUPPORT IS NOT INCLUDED IN SUPERVISOR.

Issued By: $SNA

Explanation: The $SNA program found that FULLMSG support was not included in the supervisor generation.

System Action: The $SNA program unloads.

Programmer Response: Regenerate the supervisor to include FULLMSG support.

$SNA INVALID PARTITION NUMBER IN SNAINIT FOR PU*x*

Issued By: $SNA

Explanation: The specified PU had an invalid partition number in the SNAINIT data set on the IPL volume. SNA cannot activate the specified PU #*x* without a valid partition number.

System Action: If an SNA load command issued the message, the system flags the error and activates the other designated PUs. If PUACT issued the message, SNA flags the error and ignores the command.

Response: Edit SNAINIT on the IPL volume. See Chapter 4, "Network Operation" for the correct partition for the specified PU.

$SNA INVALID PU NUMBER

Issued By: $SNA

Explanation: The only valid PU numbers for SNA are 1, 2, 3, and 4.

System Action: SNA ignores the command.

Response: Reissue the PUACT or PUDACT command with a valid PU number.

$SNA LOAD OF $CDY PROGRAM FAILED. RC = xxx

Issued By: $SNA

Explanation: An error was detected because the EDL LOAD instruction for the Communication Common Services support failed.

System Action: The $SNA program unloads.

Response: If the return code is −2, then the initialization failed for the Communication Common Services. Refer to the system console for a previous message indicating the initialization error. Refer to the *Messages and Codes* manual for an explanation of the programmer response associated with the message. For any other return code refer to the *Messages and Codes* manual for a description of the load error. Correct the error, and reload the $SNA program.

$SNA LOAD OF $NETx FAILED. RETURN CODE = xxx

Issued By: $SNA

Explanation: The activation of PU #*x* failed because the EDL LOAD instruction for the PU control blocks failed with the indicated return code.

System Action: None

Response: Look in the *Messages and Codes* manual for a description of the error. Correct the error, and reactivate the PU using the PUACT attention list command.

$SNA NO STORAGE CURRENTLY AVAILABLE IN PARTITION #1 SNA DEACTIVATING

Issued By: $SNA

Explanation: $SNA cannot obtain a 1-page storage area in partition #1.

System Action: The $SNA program unloads.

Response: You must free at least 1 page of storage in partition #1 by unloading any other programs, removing data areas, or regenerating to decrease your partition #1 supervisor size. After freeing 1 page of storage, reload the $SNA program.

$SNA PU ALREADY ACTIVATED

Issued By: $SNA

Explanation: The specified PU is already active.

System Action: SNA ignores the command.

Response: None.

$SNA PU ALREADY DEACTIVATED

Issued By: $SNA

Explanation: The specified PU is already inactive.

System Action: SNA ignores the command.

Response: None.

$SNA PUDACT IGNORED - PU ALREADY DEACTIVATING

Issued By: $SNA

Explanation: The specified PU is already deactivating.

System Action: The PU continues deactivating.

Response: None.

$SNA SNA DEACTIVATING – PUDACT IGNORED

Issued By: $SNA

Explanation: SNA is deactivating as a result of a SNADACT command. All PUs are already deactivating.

System Action: SNA continues deactivating

Response: None.

$SNA SNA DEACTIVATING, PU NOT STARTED

Issued By: $SNA

Explanation: SNA is in the process of deactivating as a result of a SNADACT command. All PUs are deactivating; no new PUs can start.

System Action: SNA continues deactivating, and the PU does not activate.

Response: After SNA deactivates, you must reactivate SNA by reloading $SNA into the system. If the PU being activated does not auto-start, you must reissue the PUACT command; otherwise, the PU automatically starts when $SNA loads. SNAINIT specifies whether or not a PU auto-starts.

$SNA SWAITM SUPPORT IS NOT INCLUDED IN SUPERVISOR.

Issued By: $SNA

Explanation: The $SNA program found that SWAITM support was not included in the supervisor generation.

System Action: The $SNA program unloads.

Programmer Response: Regenerate the supervisor to include SWAITM support.

$SNA THE $SNA PROGRAM WITH EXTENDED DATA LINK CONTROL SUPPORT REQUIRES EDX VERSION 6.0 OR LATER.

Issued By: $SNA

Explanation: The $SNA program with extended data link control support must run on an EDX supervisor level 6.0 or later.

System Action: The $SNA program unloads.

Programmer Response: Install EDX Version 6.0 or later supervisor support on your Series/1.

$SNA TIMER SUPPORT IS NOT INCLUDED IN SUPERVISOR.

Issued By: $SNA

Explanation: The $SNA program found that timer support was not included in the supervisor generation.

System Action: The $SNA program unloads.

Programmer Response: Regenerate the supervisor to include timer support (EDXTIMER or EDXTIMR2).

Appendix B. SNA Sample Application to IMS/VS

This appendix presents a Series/1 SNA sample application called SNAIMS. The application communicates with the IMS/VS manufacturing industry sample application. The *IMS/VS Installation Guide* discusses the IMS/VS sample application. The Series/1 SNA sample application presented here illustrates:

- Establishing a session
- Using SLU type P formats and protocols to send transaction names and receive output
- IMS/VS message switching
- Terminating a session.

Note: The Series/1 SNA sample application presented here is not necessarily a typical SNA application that you would write but rather an aid in diagnosing and checking out generation and connection problems to IMS/VS.

This section explains how to set up and execute the IMS/VS and Series/1 application.

IMS/VS Installation

Procedures for installation of IMS/VS and the IMS/VS sample application are discussed in Appendix B of the *IMS/VS Installation Guide*. You must modify those procedures to (1) change the terminal definition from type 2740 to SLU type P and (2) make changes necessary for the use of VTAM.

The IMS/VS system definition statements in Appendix B contain five statements that define a 2740 terminal: LINEGRP, LINE, TERMINAL, and two NAME statements. These statements must be replaced by a set of the following four macro statements that define a VTAM terminal of type SLUTYPEP: COMM, TYPE, TERMINAL, and NAME.

The APPLID parameter of the COMM macro specifies the application ID for IMS/VS. This ID must match the application ID specified on the ISAPPID parameter of the NETHOST statement in the Series/1 SNA application. The ISAPPID parameter tells the Series/1 SNA support what application ID to use when constructing INIT-SELF commands.

The TYPE macro should specify UNITYPE=SLUTYPEP.

The NAME parameter of the TERMINAL macro specifies the name of the physical terminal. This name must match the name of an LU defined in the VTAM/NCP system definition.

The MODETBL parameter specifies the name of a mode table entry that specifies SLU type P BIND parameters. This entry can be in a mode table pointed to by a PU or LU statement in the VTAM/NCP system definition, or it can be in a default table. If you do not code MODETBL, the INIT-SELF sent by the Series/1 can specify the entry name.

The OPTIONS parameter specified on the TERMINAL macro affects the protocols used between the Series/1 and IMS/VS. The NOBID option should be selected because the SNAIMS sample application does not support the BID command.

The second subparameter of the COMPTn parameter should specify BASIC as the type of editing to be used by IMS/VS. This is because SNAIMS does not support Distributed Presentation Management (DPM).

The NAME macro specifies the logical terminal name associated with the physical terminal. You can use this logical terminal name to specify the Series/1 destination for message switches.

The 2740 system definition statements define the 2740 to be the IMS/VS Master Terminal. Since an SLU type P terminal cannot be an IMS/VS Master Terminal, you must use a terminal other than the Series/1 for this purpose.

If the number of messages received at one time at a Series/1 LU is greater than the number of the LU receive buffers, the session terminates. To ensure that the session does not terminate, you can make the receive pacing count less than the number of receive buffers. The Series/1 application defines two receive buffers. Therefore, the VTAM/NCP definition should specify that the pacing count from the NCP to the Series/1 LU is 1.

Requirements of the Series/1 Sample Application

The Series/1 SNA sample application that communicates with IMS/VS is called **SNAIMS**. **SNAIMS** comes to you on diskette in object form from Information Software Distribution.

SNAIMS assumes that you use an IBM 4979 display station. The program uses lines 10 through 24 of the 4979 display screen. If you use the 4979 as both the system operator station and the terminal for SNAIMS, you may wish to redefine the display lines used so that the display areas used by the system and the application do not overlap. See the description of the IOCB statement in the *Language Reference* for information on changing the screen attributes.

The Series/1 SNA network should be capable of handling request/response units 256 bytes in length. To accomplish this, the BUFSIZ parameter value of the SNAPU configuration statement should be at least 256.

Executing the Sample Application

Before transactions can be executed from the Series/1, you must activate all of the following:

- IMS/VS
- IMS/VS sample application
- VTAM
- NCP
- Series/1 SNA support
- Series/1 sample application.

The *IMS/VS Installation Guide* discusses the procedures for activating IMS/VS and the IMS/VS sample application. Also, you must start VTAM and the NCP. If the

VTAM network does not activate the Series/1 PU and LU automatically,the VTAM network operator must vary them online.

To activate the Series/1 SNA support, you must enter the following operator command at the Series/1 operator station:

```
$L $SNA
```

Note: Make sure that you also specify $NET1 to be auto-started in SNAINIT.

Link-level communications should be established with the NCP, and VTAM should activate the first Series/1 PU and LU. If any problems arise establishing communications, refer to the debugging suggestions described in Appendix H, "Series/1 Network Activation Procedure and Checklist."

To start the SNAIMS sample application, enter the following command at your Series/1 terminal:

```
$L $SNAIMS
```

After you enter this command, SNAIMS begins execution. SNAIMS issues a NETINIT instruction to establish an LU-to-LU session. An INIT-SELF request goes to VTAM specifying **IMS** as the application with which SNAIMS wishes to communicate. SNAIMS instructs the host to send to the Series/1 a BIND that specifies SLU type P BIND parameters. Series/1 SNA support should send a positive response to BIND, thus establishing the session. The VTAM command SDT, optionally preceded by STSN, is processed.

If the processing described in the previous paragraph successfully completes, SNAIMS issues the following prompt:

```
SESSION STARTED WITH SAMPLE IMS PROGRAM
ENTER: 'HELP', 'END' OR 'TRANSACTION'
```

You execute a transaction by entering a transaction code and any transaction operands and then pressing the ENTER key at your terminal.

Appendix B of the *IMS/VS Installation Guide* shows the exact input and output formats for various transaction entries.

If you enter **HELP**, SNAIMS displays a menu that lists the valid transactions and sample operands.

If you enter **END**, the SNAIMS application terminates.

Summary of the Series/1 Sample Application

The only request units sent by the SNAIMS sample application are transactions and message switches. SNAIMS uses a single transmission (single element chain) for these requests, although IMS/VS allows them to be sent as elements of multiple transmissions.

SNAIMS sends message switches as a logical terminal name followed by a blank and text.

SNAIMS sends transactions as a transaction code followed by a blank and any operands to the IMS/VS Manufacturing Industry Application. SNAIMS does not use SLU type P message headers on input transactions. If your application uses Message Format Service (with or without DPM), your application should support message headers, since they invoke MFS and supply a Message Input Descriptor (MID) name used for formatting the input message. As an alternative, you can invoke input formatting by including an MID name in the beginning of the message text with the required MFS escape characters (//).

Request units sent by SNAIMS have both the Begin-Bracket and End-Bracket VTAM indicators set. SLU type P protocols allow End-Bracket to be omitted if Change-Direction is set, but IMS/VS always ends the bracket with a null request unit.

Except for VTAM commands, the only request units that SNAIMS can receive are message switches and output data from the IMS/VS sample application.

SNAIMS writes message switches from IMS/VS to the display station after removal of the SLU type P message header.

SNAIMS receives the output data as elements of a multiple transmission. The first transmission of the output message contains an SLU type P message header; SNAIMS discards the message header. The remaining elements of the output transmission each represent a single display line of data from the IMS/VS application. SNAIMS writes these lines to the Series/1 display as received.

Note that this example illustrates SLU type formats and protocols but does illustrate the use of DPM. The IMS/VS application constructs the output, including headers, and writes the output line by line. The IMS/VS application does not use MFS to add headers to transaction-variable data to form the finished output, nor does it use DPM to transmit only transaction-variable data.

In order for MFS output formatting to occur, a Message Output Descriptor (MOD) must be associated with the output message. The ways the MODs are associated with output messages are discussed in the *IMS/VS Programming Guide for Remote SNA Systems.*

SNAIMS

This sample Series/1 SNA application does the following:

1. Establishes a session with the host program.
2. Allows you to enter a transaction code, part number, and operands to inquire into the IMS data base.
3. Closes the session with the host program and terminates the sample program when you enter **END**.
4. Displays a menu to inform you of options when you enter **HELP**.

If an Event Driven Executive system function has an error, it displays the following message:

```
          AAAAAAAA  NETXXXX  RC= RRRR
```

AAAAAAAA is the description of the Event Driven Executive SNA error, **XXXX** is the function, and **RRRR** is the function return code.

```
*************************************************************************
*                                                                       *
*         SNAIMS                                                        *
*                                                                       *
*    THIS SAMPLE SERIES/1 SNA APPLICATION COMMUNICATES WITH             *
*    THE HOST IMS SAMPLE APPLICATION.  THE APPLICATION TO               *
*    WHICH THIS PROGRAM COMMUNICATES IS DESCRIBED IN DETAIL             *
*    IN THE IMS INSTALLATION GUIDE. (SH20-9081 APPENDIX B)              *
*                                                                       *
*                                                                       *
*************************************************************************
*        PROGRAM STATEMENT, ENTRIES AND EXTERNALS                       *
*************************************************************************
         SPACE 5
SNAIMS   PROGRAM SNA,400
         SPACE
         ENTRY   QECIND,SIGIND,BIDIND,NODATA,SNAIMS,LUSTAT
         ENTRY   TERMIND,NACKIND,RQDIND,RTSIND,HAST
         ENTRY   CANIND,EOTIND,LUNUM,RESP,RSPLNG,STATIND
         ENTRY   CDIND,FMHIND,RCCHK,STATUS,ENDADDR
         SPACE
         EXTRN   CRCODES,IRCODES,TRCODES,GETSTAT
         EXTRN   PRCODES,GRCODES,HELPOPT,DISPMSG
         SPACE
HOST     NETHOST ISAPPID=IMS,ISMODE=SAIMS0
*************************************************************************
*        ALLOCATE RESYNC DATA SET, IF ALREADY ALLOCATED EDX WILL        *
*        RETURN WITH A GOOD RETURN CODE                                 *
*************************************************************************
SNA      EQU *
         LOAD $DISKUT3,LISTPTR1,EVENT=$DISKUT3,PART=ANY
         WAIT $DISKUT3
         IF (SNAIMS,NE,-1)                              ?OPERATION OK
```

```
            MOVE RC,SNAIMS                          SAVE RETURN CODE
            PRINTEXT DU3ERR,SKIP=1          DISP DISKUT3 ERROR MESSAGE
            PRINTNUM RC,MODE=HEX                DISP ERROR RETURN CODE
            GOTO PGMEND                             TERMINATE PROGRAM
         ENDIF
*************************************************************************
*        ISSUE NETINIT INSTRUCTION TO START SESSION                     *
*************************************************************************
INIT     EQU *
         NETINIT LU=LUNUM,                                              C
               SESSPRM=PARMS,                                           C
               ATTNEV=EVENT,                                            C
               RDSCB=DSA,                                               C
               HOSTID=HOST,                                             C
               ERRCODE=SNARC,                                           C
               ACQUIRE=YES,                                             C
               RESYNC=INIT,                                             C
               RTYPE=DISK,                                              C
               EXIT=EXI
EXI      EQU *
         TCBGET LUNUM,$TCBCO2                            SAVE LUNUMBER
         IF    (SNAIMS,NE,-1)                  RETURN CODE IS NOT -1?
               MOVE RCCHK,SNAIMS                   SAVE RETURN CODE
               CALL IRCODES                      PROCESS RETURN CODE
         ENDIF
         IF    (TERMIND,EQ,-1),GOTO,TERM                 TERMINATE ?
*************************************************************************
*        ISSUE STARTING MSG FOR THE APPLICATION                         *
*************************************************************************
BEGIN    EQU *
         PRINTEXT TXT1,SKIP=1                            DISPLAY MSG
*************************************************************************
*        ISSUE PROMPT TO SCREEN, END, HELP, OR TRANSACTION              *
*************************************************************************
USPMTS   PRINTEXT PMPT1,SKIP=1,MODE=LINE
         MOVE  TRAN,TRAN1                    CLEAR BUFFER TO BLANKS
         READTEXT TRAN,SKIP=1,MODE=LINE
         IF    (TRAN,EQ,END,3),GOTO,TERM             TERMINATE?
         IF    (TRAN,EQ,HELP,4)                          HELP?
               CALL HELPOPT
               GOTO USPMTS
         ENDIF
         IF    (TRAN,EQ,BLANK,1),GOTO,USPMTS    CK FIRST CHAR FOR BLNK
*************************************************************************
*        ISSUE NETPUT TO SEND TRANSACTION TO IMS                        *
*************************************************************************
PUT      EQU *
         NETPUT LU=LUNUM,                                               C
               BUFF=TRAN,                                               C
               BYTES=TLEN,                                              C
               INVITE=YES,                                              C
               LAST=YES,                                                C
               VERIFY=YES
         IF    (SNAIMS,NE,-1)                  RETURN CODE IS NOT -1?
               MOVE RCCHK,SNAIMS                   SAVE RETURN CODE
               CALL PRCODES                      PROCESS RETURN CODE
         ENDIF
*************************************************************************
*        GET AND PROCESS STATUS INFO IF REQUIRED                        *
```

```
*************************************************************************
         IF (STATIND,EQ,-1)                          ?IS STATUS IND ON
             CALL GETSTAT                                  GET STATUS
             CALL CRCODES                 PROCESS STATUS, RETURN CODE
         ENDIF
         IF (TERMIND,EQ,-1),GOTO,TERM                ?TERMINATE IND ON
*************************************************************************
*        ISSUE NETGET TO GET RESPONSE FROM IMS                          *
*************************************************************************
         WAIT  EVENT
GET      EQU *
         NETGET LU=LUNUM,                      GET RESPONSE FROM IMS C
               BUFF=RESP,                            RECEIVE BUFFER C
               BYTES=RSPLEN,                       REC DATA LENGTH C
               RECLEN=RSPLNG                       REC BUFF LENGTH
         IF    (SNAIMS,NE,-1)                 RETURN CODE IS NOT -1?
               MOVE RCCHK,SNAIMS                   SAVE RETURN CODE
               CALL GRCODES                      PROCESS RETURN CODE
         ENDIF
         RESET EVENT
         IF (TERMIND,EQ,-1),GOTO,TERM                     TERMINATE?
*************************************************************************
*        GET AND PROCESS STATUS INFORMATION IF REQUIRED                 *
*************************************************************************
         IF (STATIND,EQ,-1)                          ?IS STATUS IND ON
             CALL GETSTAT                                  GET STATUS
             CALL CRCODES                 PROCESS STATUS, RETURN CODE
         ENDIF
         IF (TERMIND,EQ,-1),GOTO,TERM                ?TERMINATE IND ON
*************************************************************************
*        DISPLAY IMS RESPONSE ON USERS TERMINAL                         *
*************************************************************************
         IF (NODATA,EQ,-1)                                 MORE DATA?
            MOVE NODATA,0
         ELSE
         CALL  DISPMSG                             DISP IMS RESPONSE
         ENDIF
*************************************************************************
*        DETERMINE NEXT OPERATION                                       *
*************************************************************************
DNO      EQU *
         IF    (RQDIND,EQ,-1)              ?REQUEST DEFINITE RESPONSE
               MOVE RQDIND,0                 RESET DEFINITE RESP IND
               NETCTL LU=LUNUM,TYPE=ACCEPT        ISSUE POSITIVE ACK
               MOVE RCCHK,SNAIMS                   SAVE RETURN CODE
               IF (RCCHK,NE,-1)                            ?RC = -1
                  CALL CRCODES                   PROCESS RETURN CODE
                  IF (STATIND,EQ,-1)              ?IS STATUS IND ON
                     CALL GETSTAT                          GET STATUS
                     CALL CRCODES          PROCESS STATUS, RETURN CODE
                  ENDIF
                  IF (TERMIND,EQ,-1),GOTO,TERM             TERMINATE?
               ENDIF
         ENDIF
         IF    (EOTIND,EQ,-1)                    ?END OF TRANSACTION
               IF (RTSIND,EQ,0)                   ?NOT RIGHT-TO-SEND
                  NETCTL LU=LUNUM,TYPE=SIG         REQ RIGHT-TO-SEND
                  MOVE RCCHK,SNAIMS                   SAVE RETURN CODE
                  IF (RCCHK,NE,-1)                            ?RC = -1
```

```
                    CALL CRCODES                  PROCESS RETURN CODE
                    IF (STATIND,EQ,-1)              ?IS STATUS IND ON
                       CALL GETSTAT                        GET STATUS
                       CALL CRCODES       PROCESS STATUS, RETURN CODE
                    ENDIF
                    IF (TERMIND,EQ,-1),GOTO,TERM          TERMINATE?
                 ELSE
                    MOVE RTSIND,-1
                 ENDIF
              ENDIF
        ENDIF
        IF    (RTSIND,EQ,-1)                          ?RIGHT-TO-SEND
              MOVE RTSIND,0                            RESET RTSIND
              MOVE EOTIND,0                            RESET EOTIND
              GOTO USPMTS                      BRANCH TO USER PROMPT
        ELSE
              GOTO GET                           DO ANOTHER NETGET
        ENDIF
*************************************************************************
*       PROGRAM TERMINATION, ISSUE NETTERM                              *
*************************************************************************
TERM    EQU *
        NETTERM LU=LUNUM
        IF    (SNAIMS,NE,-1)                RETURN CODE IS NOT -1?
              MOVE RCCHK,SNAIMS                  SAVE RETURN CODE
              CALL TRCODES                     PROCESS RETURN CODE
        ENDIF
PGMEND  PROGSTOP
*************************************************************************
*        PARMS FOR ALLOCATE AND OPEN OF RESYNC DATA SET                 *
*************************************************************************
$DISKUT3 ECB 0                                        ECB FOR DISKUT3
*
LISTPTR1 DC A(LIST1)                        LIST POINTER FOR ALLOCATE
*
LIST1   DC A(REQUEST1)                          LIST FOR ALLOCATE
        DC A(REQUEST2)                             LIST FOR OPEN
        DC F'0'
*
REQUEST1 DC F'2'                              REQUEST FOR ALLOCATE
        DC A(DSA)                                           DSCB
        DC D'4'                              ALLOCATE FOUR RECORDS
        DC F'1'                                  DSN TYPE IS DATA
*
REQUEST2 DC F'1'                                 REQUEST FOR OPEN
        DC A(DSA)                                           DSCB
        DC D'0'
        DC F'-1'
        DSCB DS#=DSA,DSNAME=RESYNC,VOLSER=EDX002
RC      DC F'0'                    SAVE AREA FOR DISKUT3 RETURN CODE
*************************************************************************
*       SNA INDICATORS                                                  *
*************************************************************************
TERMIND  DC F'0'                                  TERMINATE INDICATOR
TEXT     DC F'0'                           CURRENT ADDR OF PRINTED MSG
NACKIND  DC F'0'                                       NEG ACK IND
QECIND   DC F'0'                                 QUIESCE INDICATOR
SIGIND   DC F'0'                                  SIGNAL INDICATOR
BIDIND   DC F'0'                                HOST BID INDICATOR
```

```
CANIND   DC F'0'                                HOST CANCEL INDICATOR
EOTIND   DC F'0'                              END OF TRANSACTION IND
RQDIND   DC F'0'                           REQUEST DEFINITE RESP IND
RTSIND   DC F'0'                                  RIGHT TO SEND IND
HAST     DC F'0'                        HOST ATT TO START A TRANSACTION
NODATA   DC F'0'                                NO DATA AVAILABLE IND
CDIND    DC F'0'                                 CHANGE DIRECTION IND
FMHIND   DC F'0'                                             FMH IND
STATIND  DC F'0'                           STATUS AVAILABLE INDICATOR
LUSTAT   DC F'0'                            LU STATUS IN STATUS FIELD
*********************************************************************
*        BUFFERS, POINTERS AND CONSTANTS                            *
*********************************************************************
LUNUM    DC F'0'                                             LU NUMBER
PARMS    DC 256C'0'                                 SAVE AREA FOR BIND
RCCHK    DC F'0'                               SAVE AREA FOR RETURN CODE
TRAN     TEXT  '     ',LENGTH=80          BUFF FOR TRANSACTION,PART NUM
TRAN1    DC 80C' '                          BLANKS TO CLEAR TRANS BUFF
TLEN     DC F'80'                                        BUFFER LENGTH
RESP     DC 256C' '                        DATA BUFF FOR IMS RESPONSE
ENDADDR  DC A(*)                                ENDADDR FOR RESP BUFF
RSPLEN   DC F'256'                     LENGTH OF DEFINED INPUT BUFFER
RSPLNG   DC F'0'                         LENGTH OF DATA REC FROM IMS
STATUS   TEXT  ' ',LENGTH=6                 SAVE AREA FOR LU STATUS
SNARC    DATA 2F'0'                            EXTENDED ERROR CODE
EVENT    ECB 0
END      DC 3C'END'                          COMPARE CONST FOR END OPT
HELP     DC 3C'HELP'                        COMPARE CONST FOR HELP OPT
BLANK    DC 1C' '                        COMPARE CONST FOR NULL TRANS
*********************************************************************
*        MESSAGES AND PROMPTS                                       *
*********************************************************************
TXT1     TEXT  'SESSION STARTED WITH SAMPLE IMS PROGRAM'
DU3ERR   TEXT  'ERROR DURING OPERATION OF DISKUT3'
PMPT1    TEXT  'ENTER:''HELP'',''END'' OR ''TRANSACTION'' '
*********************************************************************
         SPACE
         ENDPROG
         END
*********************************************************************
*                                                                   *
*        HELPOPT                                                    *
*                                                                   *
*        THIS SUBROUTINE DISPLAYS ON THE USERS CONSOLE              *
*        HELP INFORMATION ABOUT TRANSACTIONS FOR THE SAMPLE         *
*        IMS PARTS PROGRAM.                                         *
*                                                                   *
*                                                                   *
*********************************************************************
         SPACE 5
         PROGRAM MAIN=NO
         SPACE
         EXTRN  SNAIMS
         SPACE
         ENTRY HELPOPT
         SPACE
         SUBROUT HELPOPT
         SPACE
*********************************************************************
```

```
*        DISPLAY HELP TEXT                                              *
*********************************************************************
         PRINTEXT HTXT1,SKIP=1,LINE=0                        DISP LINE 1
         PRINTEXT HTXT2,SKIP=1                               DISP LINE 2
         PRINTEXT HTXT3,SKIP=1                               DISP LINE 3
         PRINTEXT HTXT4,SKIP=1                               DISP LINE 4
         PRINTEXT HTXT5,SKIP=1                               DISP LINE 5
         PRINTEXT HTXT6,SKIP=1                               DISP LINE 6
         PRINTEXT HTXT7,SKIP=1                               DISP LINE 7
         PRINTEXT HTXT8,SKIP=1                               DISP LINE 8
         PRINTEXT HTXT9,SKIP=1                               DISP LINE 9
         PRINTEXT HTXT10,SKIP=1                             DISP LINE 10
         PRINTEXT HTXT11,SKIP=1                             DISP LINE 11
         PRINTEXT HTXT12,SKIP=1                             DISP LINE 12
         PRINTEXT HTXT13,SKIP=1                             DISP LINE 13
         PRINTEXT HTXT14,SKIP=1                             DISP LINE 14
         PRINTEXT HTXT15,SKIP=1                             DISP LINE 15
         RETURN
*********************************************************************
*        HELP TEXT                                                      *
*********************************************************************
HTXT1    TEXT  '   IMS SAMPLE PARTS PROGRAM FOR THE SERIES/1          '
HTXT2    TEXT  '   THE VALID TRANSACTIONS ARE:                        '
HTXT3    TEXT  'PART PARTNUMBER                                       '
HTXT4    TEXT  'DSPALLI PARTNUMBER                                    '
HTXT5    TEXT  'DSPINV PARTNUMBER                                     '
HTXT6    TEXT  'ADDPART PARTNUMBER,DESCRIPTION                        '
HTXT7    TEXT  'ADDINV PARTNUMBER,DESCRIPTION,LOCATION                '
HTXT8    TEXT  'DLETINV PARTNUMBER,LOCATION                           '
HTXT9    TEXT  'DLETPART PARTNUMBER                                   '
HTXT10   TEXT  'CLOSE PARTNUMBER                                      '
HTXT11   TEXT  'DISBURSE PARTNUMBER,LOCATION                          '
HTXT12   TEXT  '    INITIAL PARTNUMBERS IN THE DATA BASE ARE         '
HTXT13   TEXT  'AN960C10       3003806         3007228                '
HTXT14   TEXT  '3013412        652799          7438995P002            '
HTXT15   TEXT  '7618032P101    922399-001      821250869              '
         SPACE
         ENDPROG
         END
*********************************************************************
*                                                                     *
*        DISPMSG                                                      *
*                                                                     *
*        THIS SUBROUTINE FORMATS THE INPUT DATA FROM IMS AND WRITES   *
*        IT OUT TO THE DISPLAY                                        *
*                                                                     *
*********************************************************************
         SPACE 5
         PROGRAM MAIN=NO
         SPACE
         EXTRN  SNAIMS,RESP,RSPLNG,FMHIND,ENDADDR,EOTIND
         SPACE
         ENTRY  DISPMSG
         SPACE
         SUBROUT DISPMSG
         SPACE
*********************************************************************
*        SET UP INPUT AND DISPLAY BUFFER POINTERS                     *
*********************************************************************
```

```
         MOVEA ENDADDR,RESP                    GET START ADDR OF BUFF
         ADD   ENDADDR,RSPLNG                    ADD RECV DATA LENTH
         IF   (FMHIND,EQ,-1)                             ?FMH IND ON
              MOVE FMHIND,0                            RESET FMH IND
              MOVEA #1,RESP                  GET START ADDR OF BUFF
              MOVE  DISP,RESP,(1,BYTE)            GET LENGTH OF FMH
              SHIFTR DISP,8,WORD          SHIFT BYTE TO LOW ORDER POS
              ADD   #1,DISP                CALC ADDR OF FIRST DATA
         ELSE
              MOVEA #1,RESP                 SET PTR FOR INPUT BUFF
         ENDIF
         MOVE #2,DBPTR                      RESTORE DISP BUFF PTR
*********************************************************************
*        SCAN INPUT DATA AND TRANSFER IT TO THE DISPLAY BUFFER      *
*********************************************************************
LOOP1    DO  WHILE,(#1,LT,ENDADDR)          DO UNTIL END OF IN BUFF
             MOVE DBYTE,(0,#1),(1,BYTE)     GET ONE BYTE FROM BUFFER
             IF (DBYTE,NE,NLINE,BYTE)        ? BYTE IS NOT A NEWLINE
                MOVE (DBUFF,#2),DBYTE,(1,BYTE)
                ADD  #2,1                 ADD 1 TO DISP BUFF INDEX
                IF  (#2,EQ,80)                    ?DISP BUFF FULL
                    PRINTEXT DBUFF,SKIP=1                DISP DATA
                    SUB #2,#2                CLEAR INDEX TO ZERO
                    MOVE DBUFF,BLNK,(80,BYTES)     CLEAR DISP BUFF
                ENDIF
                SUB NLCNT,NLCNT                  ZERO NEWLINE CTR
             ENDIF
             IF  (DBYTE,EQ,NLINE,BYTE)          ?BYTE IS A NEWLINE
                ADD NLCNT,1                   INCREMENT NEWLINE CTR
                IF (NLCNT,EQ,1)                          ? NLCNT = 1
                   IF (#2,NE,0)                 ?DATA IN DISP BUFF
                      PRINTEXT DBUFF,SKIP=1      WRITE DISP BUFFER
                      SUB #2,#2                      CLEAR COUNTER
                      MOVE DBUFF,BLNK,(80,BYTES)       CLEAR BUFF
                   ENDIF
                ENDIF
                IF (NLCNT,GT,1)               ?MORE THAN ONE NEWLINE
                   PRINTEXT DBUFF,SKIP=1        PRINT LINE OF BLANKS
                ENDIF
             ENDIF
             ADD #1,1                           INCREMENT DATA ADDR
         ENDDO
*********************************************************************
*        COMPLETE PROCESSING OF DISP BUFFER AT TRANSACTION END      *
*********************************************************************
         IF (EOTIND,EQ,-1)                     ?END OF TRANSACTION
            IF (#2,NE,0)                      ?DATA LEFT   IN BUFFER
                PRINTEXT DBUFF,SKIP=1           DISP REMAINING DATA
                SUB #2,#2                         ZERO OUT DISP CNT
                MOVE DBUFF,BLNK,(80,BYTES)          CLEAR DISP BUFF
             ENDIF
         ENDIF
*********************************************************************
*        SAVE AND RESET BUFFER POINTER                              *
*********************************************************************
         MOVE  DBPTR,#2                     SAVE DISP BUFF INDEX
         RETURN
*********************************************************************
DBPTR    DC F'0'                       SAVE AREA FOR DISP BUFF PTR
```

```
DISP     DC F'0'                          WORK AREA FOR FMH LENGTH
DBUFF    TEXT  LENGTH=80            DISPLAY BUFFER
DBYTE    DC C' '                                BYTE COMPARE AREA
BLNK     DC 80C' '                   EIGHTY BLANKS TO CLEAR DISP BUFF
NLINE    DC X'15'                              NEW LINE CHARACTER
NLCNT    DC F'0'                       COUNT OF CONSECTIVE NEWLINES
         ENDPROG
         END
*********************************************************************
*                                                                   *
*        GETSTAT                                                    *
*                                                                   *
*        THIS SUBROUTINE ISSUES A NETCTL RECV WHEN STATUS IS        *
*        AVAILABLE AFTER A NETGET, NETPUT OR NETCTL                 *
*        INSTRUCTION.  STATUS IS IN THE FORM OF A HEX SENSE CODE.   *
*        THESE CODES ARE DOCUMENTED IN THE IMS MANUALS.             *
*                                                                   *
*********************************************************************
*        PROGRAM STATEMENT, ENTRIES AND EXTERNALS                   *
*********************************************************************
         SPACE 5
         PROGRAM MAIN=NO
         SPACE
         EXTRN   SNAIMS,STATIND,STATUS,RCCHK,LUNUM,LUSTAT
         SPACE
         ENTRY   GETSTAT
         SPACE
         SUBROUT GETSTAT
         SPACE
*********************************************************************
*        ISSUE NETCTL TO RECEIVE STATUS                             *
*********************************************************************
         MOVE STATIND,0                              RESET STATUS IND
         MOVE LUSTAT,-1                                SET LUSTAT IND
         NETCTL LU=LUNUM,                  LU NUMBER FOR THIS SESSION C
               BUFF=STATUS,                   BUFF TO REC STATUS INFO C
               TYPE=RECV                              RECEIVE STATUS
         MOVE RCCHK,SNAIMS                          SAVE RETURN CODE
         RETURN
         ENDPROG
         END
*********************************************************************
*                                                                   *
*        IRCODES                                                    *
*                                                                   *
*        SUBROUTINE TO PROCESS NETINIT RETURN CODES                 *
*                                                                   *
*        INPUT   RCCHK, RETURN CODE FROM NETINIT OP.                *
*                                                                   *
*        OUTPUT  MESSAGES FOR RETURN CODES AND SNA INDICATORS       *
*                                                                   *
*                                                                   *
*********************************************************************
         SPACE 5
         PROGRAM MAIN=NO
         SPACE
         EXTRN  SNAIMS
         EXTRN  RCCHK,TERMIND
         SPACE
```

```
         ENTRY IRCODES
         SPACE
         SUBROUT IRCODES
         SPACE
         MOVEA ITEXT,IINVRC                SET UP DEFAULT RC MESSAGE
*************************************************************************
*        PROCESS POSITIVE RETURN CODES                                  *
*************************************************************************
         IF (RCCHK,GT,00)                       ?RETURN CODE POSITIVE
             IF (RCCHK,EQ,081)                                 ?RC=81
                 MOVEA ITEXT,IRC081          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,049)                                 ?RC=49
                 MOVEA ITEXT,IRC049          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,032)                                 ?RC=32
                 MOVEA ITEXT,IRC032          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,019)                                 ?RC=19
                 MOVEA ITEXT,IRC019          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,017)                                 ?RC=17
                 MOVEA ITEXT,IRC017          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,004)                                 ?RC=04
                 MOVEA ITEXT,IRC004          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,002)                                 ?RC=02
                 MOVEA ITEXT,IRC002          SET UP MSG FOR RC
             ENDIF
         ENDIF
*************************************************************************
*        PROCESS NEGATIVE RETURN CODES                                  *
*************************************************************************
         IF (RCCHK,LT,-1)                  ?RETURN CODE LESS THAN -1
             MOVE TERMIND,-1                    SET TERMINATE IND ON
             IF (RCCHK,EQ,-14)                              ?RC=-14
                 MOVEA ITEXT,IRCN14          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,-16)                              ?RC=-16
                 MOVEA ITEXT,IRCN16          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,-19)                              ?RC=-19
                 MOVEA ITEXT,IRCN19          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,-25)                              ?RC=-25
                 MOVEA ITEXT,IRCN25          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,-26)                              ?RC=-26
                 MOVEA ITEXT,IRCN26          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,-27)                              ?RC=-27
                 MOVEA ITEXT,IRCN27          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,-30)                              ?RC=-30
                 MOVEA ITEXT,IRCN30          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,-31)                              ?RC=-31
                 MOVEA ITEXT,IRCN31          SET UP MSG FOR RC
```

```
           ENDIF
         ENDIF
*************************************************************************
*        DISPLAY MESSAGE FOR RECEIVED RETURN CODE AND RETURN            *
*************************************************************************
         MOVE #1,ITEXT                      MOVE TEXT ADDR TO REG 1
         PRINTEXT (0,#1),SKIP=1            PRINT MSG FOR RETURN CODE
         SPACE
         RETURN
*************************************************************************
*                                                                       *
*        NETINIT RETURN CODE MESSAGES                                   *
*                                                                       *
*************************************************************************
IRC081   TEXT  'MSG FLOWS COLD-STRT, ?MSG TO HOST LOST NETINIT RC 81 '
*                                                                       *
IRC049   TEXT  'MSG FLOWS COLD-STRT, MSG TO HOST LOST NETINIT RC 49  '
*                                                                       *
IRC032   TEXT  'MESSAGE TO HOST LOST, NETINIT RC 32   '
*                                                                       *
IRC019   TEXT  'MSG FLOWS COLD-STRT, MSG FR HOST LOST NETINIT RC 19  '
*                                                                       *
IRC017   TEXT  'MSG FLOWS COLD-STRT, NO MSGS LOST NETINIT RC 17  '
*                                                                       *
IRC004   TEXT  'PARTIALLY PRESENTED MSG FR HOST LOST, NETINIT RC 04 '
*                                                                       *
IRC002   TEXT  'UNPRESENTED MESSAGE FROM HOST LOST, NETINIT RC 002   '
*                                                                       *
IRCN01   TEXT  'OPERATION SUCCESSFUL, NETINTI RC -1   '
*                                                                       *
IRCN14   TEXT  'SYSTEM ERROR, NETINIT RC -14   '
*                                                                       *
IRCN16   TEXT  'SESSION ABNORMALLY TERMINATED, NETINIT RC -16   '
*                                                                       *
IRCN19   TEXT  '$SNA NOT LOADED, NETINIT RC -19   '
*                                                                       *
IRCN25   TEXT  'INVALID LOGICAL UNIT NUMBER, NETINTI RC -25   '
*                                                                       *
IRCN26   TEXT  'LOGICAL UNIT ALREADY OPENED, NETINIT RC -26   '
*                                                                       *
IRCN27   TEXT  'NO LOGICAL UNIT AVAILABLE, NETINIT RC -27   '
*                                                                       *
IRCN30   TEXT  'BIND FROM HOST REJECTED, NETINIT RC -30   '
*                                                                       *
IRCN31   TEXT  'STSN ERROR, NETINIT RC -31   '
*                                                                       *
IINVRC   TEXT  'NETINIT RC X '
*                                                                       *
ITEXT    DC A(*)                           ADDR OF MSG TO BE PRINTED
*************************************************************************
         SPACE
         ENDPROG
         END
*************************************************************************
*                                                                       *
*        PRCODES                                                        *
*                                                                       *
*        SUBROUTINE TO PROCESS NETPUT RETURN CODES                      *
*                                                                       *
```

```
*        INPUT   RCCHK, RETURN CODE FROM NETPUT OP.                    *
*                                                                      *
*        OUTPUT  MESSAGES FOR RETURN CODES AND SNA INDICATORS          *
*                                                                      *
*                                                                      *
************************************************************************
        SPACE 5
        PROGRAM MAIN=NO
        SPACE
        EXTRN  SNAIMS,STATIND
        EXTRN  RCCHK,HAST,TERMIND
        SPACE
        ENTRY PRCODES
        SPACE
        SUBROUT PRCODES
        SPACE
        MOVEA PTEXT,PINVRC                  SET UP DEFAULT RC MESSAGE
************************************************************************
*       PROCESS POSITIVE RETURN CODES                                  *
************************************************************************
        IF (RCCHK,GT,00)                          ?RETURN CODE POSITIVE
            IF (RCCHK,EQ,001)          ?RC=01, HOST ATT TO START TRANS
                MOVE HAST,-1              SET HOST TRANSACTION IND ON
                MOVEA PTEXT,PRC001                SET UP MSG FOR RC
            ENDIF
        ENDIF
************************************************************************
*       PROCESS NEGATIVE RETURN CODES                                  *
************************************************************************
        IF (RCCHK,LT,-1)                    ?RETURN CODE LESS THAN -1
            MOVE TERMIND,-1                      SET TERMINATE IND ON
            IF (RCCHK,EQ,-09)                                 ?RC=-9
                MOVEA PTEXT,PRCN09              SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-10)                                ?RC=-10
                MOVEA PTEXT,PRCN10              SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-11)                                ?RC=-11
                MOVEA PTEXT,PRCN11              SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-12)                                ?RC=-12
                MOVEA PTEXT,PRCN12              SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-13)                                ?RC=-13
                MOVEA PTEXT,PRCN13              SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-14)                                ?RC=-14
                MOVEA PTEXT,PRCN14              SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-15)                                ?RC=-15
                MOVEA PTEXT,PRCN15              SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-16)                                ?RC=-16
                MOVEA PTEXT,PRCN16              SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-17)                                ?RC=-17
                MOVEA PTEXT,PRCN17              SET UP MSG FOR RC
                MOVE TERMIND,0                SET TERMINATE IND OFF
                MOVE STATIND,-1                     SET STATUS IND
```

```
            ENDIF
            IF (RCCHK,EQ,-18)                                   ?RC=-18
                MOVEA PTEXT,PRCN18                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-19)                                   ?RC=-19
                MOVEA PTEXT,PRCN19                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-25)                                   ?RC=-25
                MOVEA PTEXT,PRCN25                  SET UP MSG FOR RC
            ENDIF
        ENDIF
*************************************************************************
*       DISPLAY MESSAGE FOR RECEIVED RETURN CODE AND RETURN             *
*************************************************************************
        MOVE #1,PTEXT                              MOVE ADDR TO REG 1
        PRINTEXT (0,#1),SKIP=1                       DISPLAY MESSAGE
        RETURN
*************************************************************************
*                                                                       *
*       NETPUT RETURN MESSAGES                                          *
*                                                                       *
*************************************************************************
PRC001  TEXT  'HOST ATTEMPTED TO START TRANSACTION, NETPUT RC 001   '
*                                                                       *
PRCN01  TEXT  'OPERATION SUCCESSFUL, NETPUT RC -1   '
*                                                                       *
PRCN09  TEXT  'OPERATION IN PROGRESS, NETPUT RC -9   '
*                                                                       *
PRCN10  TEXT  'SESSION DOES NOT EXIST, NETPUT RC -10   '
*                                                                       *
PRCN11  TEXT  'INST/PROG NOT LINKED TO $NETCMD, NETPUT RC -11  '
*                                                                       *
PRCN12  TEXT  'INVALID LU NUMBER, NETPUT RC -12   '
*                                                                       *
PRCN13  TEXT  'INVALID REQUEST, NETPUT RC -13   '
*                                                                       *
PRCN14  TEXT  'SYSTEM ERROR, NETPUT RC -14   '
*                                                                       *
PRCN15  TEXT  'NETTERM IN PROGRESS, NETPUT RC -15   '
*                                                                       *
PRCN16  TEXT  'SESSION ABNORMALLY TERMINATED, NETPUT RC -16   '
*                                                                       *
PRCN17  TEXT  'STATUS AVAILABLE, NETPUT RC -17   '
*                                                                       *
PRCN18  TEXT  'SESSION QUIESCED, NETPUT RC -18   '
*                                                                       *
PRCN19  TEXT  '$SNA NOT LOADED, NETPUT RC -19   '
*                                                                       *
PRCN25  TEXT  'NOT RIGHT-TO-SEND, NETPUT RC -25   '
*                                                                       *
PINVRC  TEXT  'NETPUT RC X '
*                                                                       *
PTEXT   DC A(*)                    ADDR OF MSG TO BE PRINTED
*************************************************************************
        SPACE
        ENDPROG
        END
*************************************************************************
*                                                                       *
```

```
*        GRCODES                                                        *
*                                                                       *
*        SUBROUTINE TO PROCESS NETGET RETURN CODES                      *
*                                                                       *
*        INPUT   RCCHK, RETURN CODE FROM NETGET OP.                     *
*                                                                       *
*        OUTPUT  MESSAGES FOR RETURN CODES AND SNA INDICATORS           *
*                                                                       *
*                                                                       *
************************************************************************
         SPACE 5
         PROGRAM MAIN=NO
         SPACE
         EXTRN  SNAIMS,RESP,STATIND,NODATA,HAST
         EXTRN  RCCHK,EOTIND,RTSIND,TERMIND,RQDIND,FMHIND
         SPACE
         ENTRY  GRCODES
         SPACE
         SUBROUT GRCODES
         SPACE
         MOVEA GTEXT,GINVRC                 SET UP DEFAULT RC MESSAGE
************************************************************************
*        PROCESS POSITIVE RETURN CODES                                  *
************************************************************************
         IF (RCCHK,GT,00)                        ?RETURN CODE POSITIVE
             IF (RCCHK,EQ,059)                                   ?RC=59
                 MOVEA GTEXT,GRC059                SET UP MSG FOR RC
                 MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
                 MOVE RQDIND,-1       SET REQ DEFINITE RESPONSE IND ON
                 MOVE FMHIND,-1                        SET FMH IND
             ENDIF
             IF (RCCHK,EQ,058)                                   ?RC=58
                 MOVEA GTEXT,GRC058                SET UP MSG FOR RC
                 MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
                 MOVE RQDIND,-1       SET REQ DEFINITE RESPONSE IND ON
             ENDIF
             IF (RCCHK,EQ,051)                                   ?RC=51
                 MOVEA GTEXT,GRC051                SET UP MSG FOR RC
                 MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
                 MOVE FMHIND,-1                        SET FMH IND
             ENDIF
             IF (RCCHK,EQ,050)                                   ?RC=50
                 MOVEA GTEXT,GRC050                SET UP MSG FOR RC
                 MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
             ENDIF
             IF (RCCHK,EQ,047)                                   ?RC=47
                 MOVEA GTEXT,GRC047                SET UP MSG FOR RC
                 MOVE RQDIND,-1       SET REQ DEFINITE RESPONSE IND ON
                 MOVE RTSIND,-1              SET RIGHT TO SEND IND ON
                 MOVE FMHIND,-1                        SET FMH IND
             ENDIF
             IF (RCCHK,EQ,046)                                   ?RC=46
                 MOVEA GTEXT,GRC046                SET UP MSG FOR RC
                 MOVE RQDIND,-1       SET REQ DEFINITE RESPONSE IND ON
                 MOVE RTSIND,-1              SET RIGHT TO SEND IND ON
             ENDIF
             IF (RCCHK,EQ,043)                                   ?RC=43
                 MOVEA GTEXT,GRC043                SET UP MSG FOR RC
                 MOVE RQDIND,-1       SET REQ DEFINITE RESPONSE IND ON
```

```
    MOVE FMHIND,-1                          SET FMH IND
ENDIF
IF (RCCHK,EQ,042)                                   ?RC=42
    MOVEA GTEXT,GRC042                  SET UP MSG FOR RC
    MOVE RQDIND,-1        SET REQ DEFINITE RESPONSE IND ON
ENDIF
IF (RCCHK,EQ,039)                                   ?RC=39
    MOVEA GTEXT,GRC039                  SET UP MSG FOR RC
    MOVE RTSIND,-1              SET RIGHT TO SEND IND ON
    MOVE FMHIND,-1                          SET FMH IND
ENDIF
IF (RCCHK,EQ,038)                                   ?RC=38
    MOVEA GTEXT,GRC038                  SET UP MSG FOR RC
    MOVE RTSIND,-1              SET RIGHT TO SEND IND ON
ENDIF
IF (RCCHK,EQ,035)                                   ?RC=35
    MOVEA GTEXT,GRC035                  SET UP MSG FOR RC
    MOVE FMHIND,-1                          SET FMH IND
ENDIF
IF (RCCHK,EQ,034)                                   ?RC=34
    MOVEA GTEXT,GRC034                  SET UP MSG FOR RC
ENDIF
IF (RCCHK,EQ,033)                                   ?RC=33
    MOVEA GTEXT,GRC033                  SET UP MSG FOR RC
    MOVE FMHIND,-1                          SET FMH IND
ENDIF
IF (RCCHK,EQ,032)                                   ?RC=32
    MOVEA GTEXT,GRC032                  SET UP MSG FOR RC
ENDIF
IF (RCCHK,EQ,027)                                   ?RC=27
    MOVEA GTEXT,GRC027                  SET UP MSG FOR RC
    MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
    MOVE RQDIND,-1        SET REQ DEFINITE RESPONSE IND ON
ENDIF
IF (RCCHK,EQ,026)                                   ?RC=26
    MOVEA GTEXT,GRC026                  SET UP MSG FOR RC
    MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
    MOVE RQDIND,-1        SET REQ DEFINITE RESPONSE IND ON
ENDIF
IF (RCCHK,EQ,019)                                   ?RC=19
    MOVEA GTEXT,GRC019                  SET UP MSG FOR RC
    MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
    MOVE FMHIND,-1                          SET FMH IND
ENDIF
IF (RCCHK,EQ,018)                                   ?RC=18
    MOVEA GTEXT,GRC018                  SET UP MSG FOR RC
    MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
ENDIF
IF (RCCHK,EQ,015)                                   ?RC=15
    MOVEA GTEXT,GRC015                  SET UP MSG FOR RC
    MOVE RQDIND,-1        SET REQ DEFINITE RESPONSE IND ON
    MOVE RTSIND,-1              SET RIGHT TO SEND IND ON
    MOVE FMHIND,-1                          SET FMH IND
ENDIF
IF (RCCHK,EQ,014)                                   ?RC=14
    MOVEA GTEXT,GRC014                  SET UP MSG FOR RC
    MOVE RQDIND,-1        SET REQ DEFINITE RESPONSE IND ON
    MOVE RTSIND,-1              SET RIGHT TO SEND IND ON
ENDIF
```

```
            IF (RCCHK,EQ,011)                                   ?RC=11
                MOVEA GTEXT,GRC011                  SET UP MSG FOR RC
                MOVE RQDIND,-1        SET REQ DEFINITE RESPONSE IND ON
                MOVE FMHIND,-1                          SET FMH IND
            ENDIF
            IF (RCCHK,EQ,010)                                   ?RC=10
                MOVEA GTEXT,GRC010                  SET UP MSG FOR RC
                MOVE RQDIND,-1        SET REQ DEFINITE RESPONSE IND ON
            ENDIF
            IF (RCCHK,EQ,007)                                   ?RC=07
                MOVEA GTEXT,GRC007                  SET UP MSG FOR RC
                MOVE RTSIND,-1              SET RIGHT TO SEND IND ON
                MOVE FMHIND,-1                          SET FMH IND
            ENDIF
            IF (RCCHK,EQ,006)                                   ?RC=06
                MOVEA GTEXT,GRC006                  SET UP MSG FOR RC
                MOVE RTSIND,-1              SET RIGHT TO SEND IND ON
            ENDIF
            IF (RCCHK,EQ,003)                                   ?RC=03
                MOVEA GTEXT,GRC003                  SET UP MSG FOR RC
                MOVE FMHIND,-1                          SET FMH IND
            ENDIF
            IF (RCCHK,EQ,002)                                   ?RC=02
                MOVEA GTEXT,GRC002                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,001)                                   ?RC=01
                MOVEA GTEXT,GRC001                  SET UP MSG FOR RC
                MOVE FMHIND,-1                          SET FMH IND
            ENDIF
        ENDIF
*************************************************************************
*       PROCESS NEGATIVE RETURN CODES                                   *
*************************************************************************
        IF (RCCHK,LT,-1)                        RETURN CODE LESS THAN -1
            MOVE TERMIND,-1                         SET TERMINATE IND ON
            IF (RCCHK,EQ,-09)                                   ?RC=-9
                MOVEA GTEXT,GRCN09                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-10)                                  ?RC=-10
                MOVEA GTEXT,GRCN10                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-11)                                  ?RC=-11
                MOVEA GTEXT,GRCN11                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-12)                                  ?RC=-12
                MOVEA GTEXT,GRCN12                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-13)                                  ?RC=-13
                MOVEA GTEXT,GRCN13                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-14)                                  ?RC=-14
                MOVEA GTEXT,GRCN14                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-15)                                  ?RC=-15
                MOVEA GTEXT,GRCN15                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-16)                                  ?RC=-16
                MOVEA GTEXT,GRCN16                  SET UP MSG FOR RC
            ENDIF
```

```
            IF (RCCHK,EQ,-17)                                   ?RC=-17
                MOVEA GTEXT,GRCN17                  SET UP MSG FOR RC
                MOVE TERMIND,0                    SET TERMINATE IND OFF
                MOVE STATIND,-1                         SET STATUS IND
            ENDIF
            IF (RCCHK,EQ,-19)                                   ?RC=-19
                MOVEA GTEXT,GRCN19                  SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-25)                                   ?RC=-25
                MOVEA GTEXT,GRCN25                  SET UP MSG FOR RC
                MOVE TERMIND,0                        RESET TERM IND
                MOVE NODATA,-1                       SET NO DATA IND
            ENDIF
            IF (RCCHK,EQ,-26)                                   ?RC=-26
                MOVEA GTEXT,GRCN26                  SET UP MSG FOR RC
                MOVE TERMIND,0                        RESET TERM IND
                MOVE HAST,-1                SET HOST INIT TRANSACTION
            ENDIF
        ENDIF
*************************************************************************
*       DISPLAY MESSAGE FOR RECEIVED RETURN CODE                        *
*************************************************************************
        MOVE #1,GTEXT                               MOVE ADDR TO REG 1
        PRINTEXT (0,#1),SKIP=1                PRINT MSG FOR RETURN CODE
        SPACE
        RETURN
*************************************************************************
*                                                                       *
*       NETGET RETURN MESSAGES                                          *
*                                                                       *
*************************************************************************
GRC059  TEXT  'SOT, EOT, EOM, RQD, FMH RECEIVED  NETGET RC 59  '
*                                                                       *
GRC058  TEXT  'SOT, EOT, EOM, RQD, RECEIVED  NETGET RC 58  '
*                                                                       *
GRC051  TEXT  'SOT, EOT, EOM, FMH RECEIVED  NETGET RC 51  '
*                                                                       *
GRC050  TEXT  'SOT, EOT, EOM RECEIVED  NETGET RC 50  '
*                                                                       *
GRC047  TEXT  'SOT, EOT, EOM, FMH, RTS, RECEIVED NETGET RC 47  '
*                                                                       *
GRC046  TEXT  'SOT, EOM, RTS, RQD RECEIVED NETGET RC 46  '
*                                                                       *
GRC043  TEXT  'SOT, EOM, FMH, RQD RECEIVED NETGET RC 43  '
*                                                                       *
GRC042  TEXT  'SOT, EOM, RQD RECEIVED NETGET RC 42  '
*                                                                       *
GRC039  TEXT  'SOT, EOM, FMH, RTS RECEIVED NETGET 39  '
*                                                                       *
GRC038  TEXT  'SOT, EOM, RTS RECEIVED NETGET RC 38  '
*                                                                       *
GRC035  TEXT  'SOT, EOM, FMH RECEIVED NETGET RC 35  '
*                                                                       *
GRC034  TEXT  'SOT, EOM RECEIVED NETGET RC 34  '
*                                                                       *
GRC033  TEXT  'SOT, FMH  RECEIVED, NETGET RC 33  '
*                                                                       *
GRC032  TEXT  'SOT RECEIVED NETGET RC 32   '
*                                                                       *
```

```
GRC027   TEXT  'EOT, EOM, FMH, RQD RECEIVED NETGET RC 27  '
*                                                                      *
GRC026   TEXT  'EOT, EOM, RQD RECEIVED NETGET RC 26  '
*                                                                     *
GRC019   TEXT  'EOT, EOM, FMH RECEIVED NETGET RC 19  '
*                                                                      *
GRC018   TEXT  'EOT, EOM RECEIVED NETGET RC 18  '
*                                                                      *
GRC015   TEXT  'EOM, FMH, RTS, RQD, RECEIVED NETGET RC 15  '
*                                                                      *
GRC014   TEXT  'EOM, RTS, RQD RECEIVED NETGET RC 14  '
*                                                                      *
GRC011   TEXT  'EOM, FMH, RQD RECEIVED NETGET RC 11  '
*                                                                      *
GRC010   TEXT  'EOM, RQD RECEIVED  NETGET RC 10  '
*                                                                      *
GRC007   TEXT  'EOM, FMH, RTS RECEIVED NETGE RC 07  '
*                                                                      *
GRC006   TEXT  'EOM, RTS RECEIVED NETGET RC 06  '
*                                                                      *
GRC003   TEXT  'EOM, FMH RECEIVED NETGET RC 03   '
*                                                                      *
GRC002   TEXT  'EOM RECEIVED NETGET RC 02   '
*                                                                      *
GRC001   TEXT  'FMH RECEIVED NETGET RC 01   '
*                                                                      *
GRCN01   TEXT  'OPERATION SUCCESSFUL, NETGET RC -1   '
*                                                                      *
GRCN09   TEXT  'OPERATION IN PROGRESS, NETGET RC -09   '
*                                                                      *
GRCN10   TEXT  'SESSION DOES NOT EXIST, NETGET RC -10   '
*                                                                      *
GRCN11   TEXT  'INST/PROG NOT LINKED TO $NETCMD NETGET RC -11  '
*                                                                      *
GRCN12   TEXT  'INVALID LU NUMBER, NETGET RC -12   '
*                                                                      *
GRCN13   TEXT  'INVALID REQUEST, NETGET RC -13   '
*                                                                      *
GRCN14   TEXT  'SYSTEM ERROR, NETGET RC -14   '
*                                                                      *
GRCN15   TEXT  'NETTERM IN PROGRESS, NETGET RC -15   '
*                                                                      *
GRCN16   TEXT  'SESSION ABNORMALLY TERMINATED, NETGET RC -16   '
*                                                                      *
GRCN17   TEXT  'STATUS AVAILABLE, NETGET RC -17   '
*                                                                      *
GRCN19   TEXT  '$SNA NOT LOADED, NETGET RC -19   '
*                                                                      *
GRCN25   TEXT  'NO MESSAGES AVAILABLE, NETGET RC -25   '
*                                                                      *
GRCN26   TEXT  'HOST INITIATED A TRANSACTION,NETGET RC -26   '
*                                                                      *
GINVRC   TEXT  'NETGET RC X '
************************************************************************
GTEXT    DC A(*)                                ADDR FOR PRINTED MSG
         SPACE
         ENDPROG
         END
************************************************************************
```

```
*                                                                     *
*       CRCODES                                                       *
*                                                                     *
*       SUBROUTINE TO PROCESS NETCTL RETURN CODES                     *
*                                                                     *
*       INPUT   RCCHK, CONTAINS RETURN CODES FROM NETCTL OP.          *
*                                                                     *
*       OUTPUT  RETURN CODE MESSAGE AND SNA INDICATORS                *
*                                                                     *
*                                                                     *
***********************************************************************
        SPACE 5
        PROGRAM MAIN=NO
        SPACE
        EXTRN  SNAIMS,STATIND,STATUS,LUSTAT
        EXTRN  RCCHK,NACKIND,BIDIND,TERMIND
        EXTRN  CANIND,EOTIND,QECIND,SIGIND,RTSIND
        SPACE
        ENTRY  CRCODES
        SPACE
        SUBROUT CRCODES
        SPACE
        MOVEA CTEXT,CINVRC                SET UP DEFAULT RC MESSAGE
***********************************************************************
*       PROCESS POSITIVE RETURN CODES                                 *
***********************************************************************
        IF (RCCHK,GT,00)                      ?RETURN CODE POSITIVE
            IF (RCCHK,EQ,112)                                ?RC=112
                MOVEA CTEXT,CRC112              SET UP MSG FOR RC
                MOVE NACKIND,-1                SET NEG ACK IND ON
            ENDIF
            IF (RCCHK,EQ,096)                                 ?RC=96
                MOVEA CTEXT,CRC096              SET UP MSG FOR RC
                MOVE QECIND,-1               RESET QUIESCE INDICATOR
            ENDIF
            IF (RCCHK,EQ,080)                                 ?RC=80
                MOVEA CTEXT,CRC080              SET UP MSG FOR RC
                MOVE SIGIND,-1                 SET SIGNAL IND ON
            ENDIF
            IF (RCCHK,EQ,048)                                 ?RC=48
                MOVEA CTEXT,CRC048              SET UP MSG FOR RC
                MOVE TERMIND,-1               SET TERMINATE IND ON
            ENDIF
            IF (RCCHK,EQ,034)                                 ?RC=34
                MOVEA CTEXT,CRC034              SET UP MSG FOR RC
                MOVE CANIND,-1                 SET CANCEL IND ON
                MOVE RTSIND,-1               SET RIGHT-TO-SEND IND
            ENDIF
            IF (RCCHK,EQ,033)                                 ?RC=33
                MOVEA CTEXT,CRC033              SET UP MSG FOR RC
                MOVE RTSIND,-1               SET RIGHT-TO-SEND IND
                MOVE EOTIND,-1         SET END OF TRANSACTION IND ON
            ENDIF
            IF (RCCHK,EQ,032)                                 ?RC=32
                MOVEA CTEXT,CRC032              SET UP MSG FOR RC
                MOVE CANIND,-1                 SET CANCEL IND ON
            ENDIF
            IF (RCCHK,EQ,018)                                 ?RC=18
                MOVEA CTEXT,CRC018              SET UP MSG FOR RC
```

```
            MOVE RTSIND,-1                  SET RIGHT-TO-SEND IND
            MOVE LUSTAT,-1                        SET LU STATUS
        ENDIF
        IF (RCCHK,EQ,017)                                 ?RC=17
            MOVEA CTEXT,CRC017                 SET UP MSG FOR RC
            MOVE EOTIND,-1          SET END OF TRANSACTION IND ON
            MOVE LUSTAT,-1                        SET LU STATUS
        ENDIF
        IF (RCCHK,EQ,016)                                 ?RC=16
            MOVEA CTEXT,CRC016                 SET UP MSG FOR RC
            MOVE LUSTAT,-1                        SET LU STATUS
        ENDIF
        IF (RCCHK,EQ,002)                                 ?RC=02
            MOVEA CTEXT,CRC002                 SET UP MSG FOR RC
            MOVE RTSIND,-1                  SET RIGHT-TO-SEND IND
        ENDIF
        IF (RCCHK,EQ,001)                                 ?RC=01
            MOVEA CTEXT,CRC001                 SET UP MSG FOR RC
            MOVE EOTIND,-1             SET END OF TRANSACTION IND
        ENDIF
    ENDIF
*********************************************************************
*       PROCESS NEGATIVE RETURN CODES                               *
*********************************************************************
    IF (RCCHK,LT,-01)               ?RETURN CODE IS LESS THAN -1
        MOVE TERMIND,-1                     SET TERMINATE IND ON
        IF (RCCHK,EQ,-09)                                 ?RC=-9
            MOVEA CTEXT,CRCN09                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-10)                                ?RC=-10
            MOVEA CTEXT,CRCN10                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-11)                                ?RC=-11
            MOVEA CTEXT,CRCN11                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-12)                                ?RC=-12
            MOVEA CTEXT,CRCN12                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-13)                                ?RC=-13
            MOVEA CTEXT,CRCN13                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-14)                                ?RC=-14
            MOVEA CTEXT,CRCN14                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-15)                                ?RC=-15
            MOVEA CTEXT,CRCN15                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-16)                                ?RC=-16
            MOVEA CTEXT,CRCN16                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-17)                                ?RC=-17
            MOVEA CTEXT,CRCN17                 SET UP MSG FOR RC
            MOVE TERMIND,0                  SET TERMINATE IND OFF
            MOVE STATIND,-1                    SET STATUS IND ON
        ENDIF
        IF (RCCHK,EQ,-18)                                ?RC=-18
            MOVEA CTEXT,CRCN18                 SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-19)                                ?RC=-19
```

```
            MOVEA CTEXT,CRCN19                    SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-25)                                  ?RC=-25
            MOVEA CTEXT,CRCN25                    SET UP MSG FOR RC
        ENDIF
        IF (RCCHK,EQ,-26)                                  ?RC=-26
            MOVEA CTEXT,CRCN26                    SET UP MSG FOR RC
        ENDIF
    ENDIF
*************************************************************************
*       DISPLAY MESSAGE FOR RECEIVED RETURN CODE                        *
*************************************************************************
    MOVE #1,CTEXT                              MOVE ADDR TO REG 1
    PRINTEXT (0,#1),SKIP=1                PRINT MSG FOR RETURN CODE
    IF (LUSTAT,EQ,-1)                                 ? LU STATUS
        PRINTNUM STATUS,MODE=HEX                     PRINT STATUS
        MOVE LUSTAT,0                          RESET LU STATUS IND
    ENDIF
    RETURN
*************************************************************************
*                                                                       *
*       NETCTL RETURN CODE MESSAGES                                     *
*                                                                       *
*************************************************************************
CRC112  TEXT  'NEGATIVE RESPONSE RECEIVED, NETCTL RC 112   '
*                                                                       *
CRC096  TEXT  'RELEASE QUIESCED RECEIVED, NETCTL RC 96   '
*                                                                       *
CRC080  TEXT  'SIGNAL RECEIVED, NETCTL RC 80   '
*                                                                       *
CRC048  TEXT  'SHUTDOWN RECEIVED, NETCTL RC 48   '
*                                                                       *
CRC034  TEXT  'CANCEL WITH CHANGE DIRECTION RECEIVED, NETCTL RC 34  '
*                                                                       *
CRC033  TEXT  'CANCEL WITH END TRANSACTION RECEIVED, NETCTL RC 33   '
*                                                                       *
CRC032  TEXT  'CANCEL RECEIVED, NETCTL RC 32   '
*                                                                       *
CRC018  TEXT  'LUSTAT WITH CHANGE DIRECTION RECEIVED, NETCTL RC 18  '
*                                                                       *
CRC017  TEXT  'LUSTAT WITH WITH EOT RECEIVED, NETCTL RC 17  '
*                                                                       *
CRC016  TEXT  'LUSTAT RECEIVED, NETCTL RC 16   '
*                                                                       *
CRC002  TEXT  'CHANGE DIRECTION RECEIVED, NETCTL 02   '
*                                                                       *
CRC001  TEXT  'END TRANSACTION RECEIVED, NETCTL RC 01   '
*                                                                       *
CRCN01  TEXT  'OPERATION SUCCESSFUL, NETCTL RC -1   '
*                                                                       *
CRCN09  TEXT  'OPERATION IN PROGRESS, NETCTL RC -9   '
*                                                                       *
CRCN10  TEXT  'SESSION DOES NOT EXIST, NETCTL RC -10   '
*                                                                       *
CRCN11  TEXT  'INST/PROG NOT LINKED TO $NETCMD NETCTL RC -11  '
*                                                                       *
CRCN12  TEXT  'INVALID LU NUMBER, NETCTL RC -12   '
*                                                                       *
CRCN13  TEXT  'INVALID REQUEST, NETCTL RC -13   '
```

```
*                                                                     *
CRCN14   TEXT  'SYSTEM ERROR, NETCTL RC -14   '
*                                                                     *
CRCN15   TEXT  'NETTERM IN PROGRESS, NETCTL RC -15   '
*                                                                     *
CRCN16   TEXT  'SESSION ABNORMALLY TERMINATED, NETCTL RC -16   '
*                                                                     *
CRCN17   TEXT  'STATUS AVAILABLE, NETCTL RC -17   '
*                                                                     *
CRCN18   TEXT  'SESSION QUIESCED, NETCTL RC -18   '
*                                                                     *
CRCN19   TEXT  '$SNA NOT LOADED, NETCTL RC -19   '
*                                                                     *
CRCN25   TEXT  'NOT RIGHT-TO-SEND, NETCTL RC -25   '
*                                                                     *
CRCN26   TEXT  'NO STATUS AVAILABLE, NETCTL RC -26   '
*                                                                     *
CINVRC   TEXT  'NETCTL RC X '
***********************************************************************
CTEXT    DC A(*)                                  ADDR OF PRINTED MSG
         SPACE
         ENDPROG
         END
***********************************************************************
*                                                                     *
*        TRCODES                                                      *
*                                                                     *
*        SUBROUTINE TO PROCESS NETTERM RETURN CODES                   *
*                                                                     *
*        INPUT   RCCHK, CONTAINS RETURN CODE FROM NETTERM OP.         *
*                                                                     *
*        OUTPUT  MESSAGES FOR RETURN CODES AND SNA INDICATORS         *
*                                                                     *
***********************************************************************
         SPACE 5
         PROGRAM MAIN=NO
         SPACE
         EXTRN  SNAIMS
         EXTRN  RCCHK
         SPACE
         ENTRY TRCODES
         SPACE
         SUBROUT TRCODES
         SPACE
         MOVEA TTEXT,TINVRC                  SET UP DEFAULT MESSAGE
***********************************************************************
*        PROCESS POSITIVE RETURN CODES                                *
***********************************************************************
         IF (RCCHK,GT,00)                   ?RETURN CODE POSITIVE
             IF (RCCHK,EQ,09)                                 ?RC=9
                 MOVEA TTEXT,TRC009          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,08)                                 RC=8?
                 MOVEA TTEXT,TRC008          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,07)                                 ?RC=7
                 MOVEA TTEXT,TRC007          SET UP MSG FOR RC
             ENDIF
             IF (RCCHK,EQ,06)                                 ?RC=6
```

```
                MOVEA TTEXT,TRC006                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,05)                                    ?RC=5
                MOVEA TTEXT,TRC005                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,04)                                    ?RC=4
                MOVEA TTEXT,TRC004                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,03)                                    ?RC=3
                MOVEA TTEXT,TRC003                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,02)                                    ?RC=2
                MOVEA TTEXT,TRC002                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,01)                                    ?RC=1
                MOVEA TTEXT,TRC001                SET UP MSG FOR RC
            ENDIF
        ENDIF
*************************************************************************
*       PROCESS NEGATIVE RETURN CODES                                   *
*************************************************************************
        IF (RCCHK,LT,-1)                  ?RETURN CODE LESS THAN -1
            IF (RCCHK,EQ,-10)                                  ?RC=-10
                MOVEA TTEXT,TRCN10                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-11)                                  ?RC=-11
                MOVEA TTEXT,TRCN11                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-12)                                  ?RC=-12
                MOVEA TTEXT,TRCN12                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-14)                                  ?RC=-14
                MOVEA TTEXT,TRCN14                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-15)                                  ?RC=-15
                MOVEA TTEXT,TRCN15                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-16)                                  ?RC=-16
                MOVEA TTEXT,TRCN16                SET UP MSG FOR RC
            ENDIF
            IF (RCCHK,EQ,-19)                                  ?RC=-19
                MOVEA TTEXT,TRCN19                SET UP MSG FOR RC
            ENDIF
        ENDIF
*************************************************************************
*       DISPLAY MESSAGE FOR RECEIVED RETURN CODE AND RETURN             *
*************************************************************************
        MOVE #1,TTEXT               MOVE ADDR TO REG 1
        PRINTEXT (0,#1),SKIP=1      PRINT MSG FOR RETURN CODE
        RETURN
*************************************************************************
*                                                                       *
*         NETTERM RETURN CODE MESSAGES                                  *
*                                                                       *
*************************************************************************
TRC009   TEXT  'CANCEL RECEIVED , -RSP SENT  NETTERM RC 09  '
*                                                                       *
TRC008   TEXT  'CANCEL RECEIVED DURING NETTERM, NETTERM RC 08   '
*                                                                       *
```

```
TRC007   TEXT  'CANCEL ,-RSP RECEIVED, -RSP SENT NETTERM RC 07  '
*                                                                      *
TRC006   TEXT  'CANCEL ,-RSP RECEIVED  NETTERM RC 06  '
*                                                                      *
TRC005   TEXT  'CANCEL AND -RSP SENT   NETTERM RC 05   '
*                                                                      *
TRC004   TEXT  'CANCEL SENT  NETTERM RC 04   '
*                                                                      *
TRC003   TEXT  '-RSP RECEIVED AND SENT  NETTERM RC 03 '
*                                                                      *
TRC002   TEXT  '-RSP RECEIVED  NETTERM RC 02   '
*                                                                      *
TRC001   TEXT  '-RSP SENT  NETTERM RC 01   '
*                                                                      *
TRCN01   TEXT  'OPERATION SUCCESSFUL, NETTERM RC -1   '
*                                                                      *
TRCN10   TEXT  'SESSION DOES NOT EXIST, NETTERM RC -10   '
*                                                                      *
TRCN11   TEXT  'INST/PROG NOT LINKED TO $NETCMD, NETTERM RC -11  '
*                                                                      *
TRCN12   TEXT  'INVALID LU NUMBER, NETTERM RC -12   '
*                                                                      *
TRCN14   TEXT  'SYSTEM ERROR, NETTERM RC -14   '
*                                                                      *
TRCN15   TEXT  'NETTERM IN PROGRESS, NETTERM RC -15   '
*                                                                      *
TRCN16   TEXT  'SESSION ABNORMALLY TERMINATED, NETTERM RC -16   '
*                                                                      *
TRCN19   TEXT  '$SNA NOT LOADED, NETTERM RC -19   '
*                                                                      *
TINVRC   TEXT  'NETTERM RC X '
*                                                                      *
TTEXT    DC A(*)                                  ADDR OF MSG TO PRINT
************************************************************************
         SPACE
         ENDPROG
         END
```

Appendix C. VTAM Considerations

This appendix contains a list of VTAM programming considerations that apply when you use Series/1 SNA support:

XID

In a VTAM Release 2 network, you must use the short form (six bytes) of the Exchange Station Identification (XID) command on the Series/1 SNAPU configuration statement for switched line protocol. In a VTAM network beyond Release 2, you can use either the short form or the long form (twenty bytes) of the XID command. This value is defined to VTAM in the PU definition. (See Appendix F, "Data Area Descriptions" for the XID formats.)

Station address

You must also define the secondary station address for the SDLC adapter card and the VTAM PU definition. Code the VTAM ADDR parameter to do this.

When using the SDLC support, the secondary station address is jumpered on the card.

When using shared SDLC, the secondary station address may or may not be jumpered on the card. The secondary station address may be specified by the Network Definition Utility SDLC station configuration record and the SNAPU SECSTAT parameter. If using shared SDLC hardware address recognition, the secondary station address must be jumpered on the card. The SECSTAT parameter will be "FF", and the NDU SDLC station configuration record must have STN1 = FF. If using shared SDLC software address recognition, the SECSTAT parameter specifies the secondary station address used and must be the same value as the STNx parameter on the NDU SDLC station configuration record

Block number

You must specify the Series/1 block number X'021' in the VTAM PU macro. (The block number is defined in the XID format in Appendix F, "Data Area Descriptions."

Pacing

Outbound pacing is not supported in a VTAM Release 2 network. VTAM definitions (PU and LU statements) or the VTAM mode table entry define pacing values. The only time a primary application can specify a pacing inbound value is in a VTAM network beyond Release 2.

Series/1 SNA NETHOST Statement

The VTAM Release 2 network does not support the ISQUEUE parameter of the Series/1 NETHOST statement.

Host network deactivation

When the Series/1 is on a leased line, VTAM deactivates the PU under the following conditions: -

- All the defined LUs had LU to LU sessions, and all sessions terminate. In this case, if you do not want VTAM to deactivate the PU, code the VTAM PU macro specifying the DISCNT=NO parameter. (For more information, see the appropriate VTAM programmer's guide.)
- The Series/1 deactivates its network via a SNADACT operator command. If you do not want VTAM to deactivate the PU, do not let the Series/1 network deactivate. That is, activate the network by a $L $SNA command and do not issue a SNADACT or PUDACT command until all processing is complete.

For a switched communications line, only the physical connection is broken. The host can be called again when the Series/1 network is reactivated.

Because you deactivate a session when you send the response to UNBIND, you can activate another session (NETINIT issued) immediately afterwards. If the new session requests that an INIT-SELF (ACQUIRE=YES) be sent and the VTAM program's CLSDST for the previous session has not completed, VTAM sends a NSPE to the Series/1 and a return code of −14 (system error) returns from NETINIT. This terminates the activation process of the new session. You can retry the NETINIT until it is successful. This condition occurs only when both sessions use the same LU.

Appendix D. Host Subsystem Considerations

You can use Series/1 SNA support to allow the Series/1 PU to function as a 3790-type controller in an Information Management System/Virtual Storage (IMS/VS) environment. Series/1 SNA support conforms to the formats and protocols defined by IMS/VS for a type "P" secondary LU (SLU-P).

Series/1 SNA support also allows a Series/1 physical unit to function as a 3790-type controller in a Customer Information Control System/Virtual Storage (CICS/VS) environment. The Series/1 SNA support conforms to the formats and protocols defined by CICS/VS for the "full function" secondary LU.

Series/1 SNA support also allows a Series/1 physical unit to function as a 3650-type controller in a Customer Information Control System (CICS/VS) environment. The Series/1 SNA bracket protocol conforms to the formats and protocols defined by CICS/VS for the "PIPELINE" secondary LU using DUPLEX.

The considerations you should take into account when using these host subsystems to communicate with the Series/1 in an SNA network appear under the following topics:

- "IMS/VS Generation"
- "Distributed Presentation Management" on page D-7
- "CICS/VS Generation" on page D-16.

IMS/VS Generation

You must describe the Series/1 LUs when defining the IMS/VS system. Each LU used by Series/1 applications in session with IMS/VS corresponds to a "physical terminal" defined to IMS/VS by the IMS/VS macros COMM, TYPE, TERMINAL, and NAME and their associated parameters.

Following is an illustration of an SLU type P physical terminal consisting of one input/output component. You can reference the component in IMS/VS applications by its logical terminal name, INOUT1.

```
COMM  APPLID=IMS,•••                                    C
TYPE  UNITYPE=SLUTYPEP,•••                              C
TERMINAL NAME=PTERM1,                                   C
      MODETBL=ENTRY1,                                   C
      COMPT1=(PROGRAM1,DPM-A1,IGNORE),                  C
      •
      •
      •
NAME  INOUT1,COMPT=1,ICOMPT=1
```

Figure D-1. IMS/VS System Definition

COMM Macro

The COMM macro specifies general communication requirements that are not associated with any particular terminal type.

The APPLID parameter specifies the VTAM application identification for IMS/VS. The name specified for this parameter must be the same as the name specified for the ISAPPID parameter on the NETHOST statement for each Series/1 program that communicates with IMS/VS. For example, if the IMS/VS system definition is:

```
COMM  APPLID=IMS,                                    C
      •
      •
      •
```

the Series/1 application is:

```
NETHOST ISAPPID=IMS,                                 C
        •
        •
        •
```

TYPE Macro

The TYPE macro defines the beginning of a set of TERMINAL and NAME macro statements. The set of macro statements defines one or more terminals of the same type.

The UNITYPE parameter specifies the terminal device type contained in this set of terminal definitions. Code the UNITYPE parameter as follows for the Series/1 in an SLU type P environment:

```
TYPE UNITYPE=SLUTYPEP
```

TERMINAL Macro

The TERMINAL macro defines physical and logical terminal characteristics. The IMS/VS concept of a physical terminal corresponds to a LU on the Series/1.

The NAME parameter specifies the physical terminal name. The name specified on the NAME parameter must match the label on an LU statement in the VTAM/NCP generation. Following is an example of specifying physical terminal names. If the IMS/VS System definition is:

```
        TERMINAL NAME=PTERM1,                                          C
                 •
                 •
                 •
```

the VTAM/NCP definition is:

```
PU1      PU    PUTYPE=2,ADDR=C1,•••                                     C
PTERM1   LU    LOCADDR=1,PACING=(2,1),•••                               C
PTERM2   LU    LOCADDR=2,PACING=(2,1),•••                               C
               •
               •
               •
```

The LTERM parameter specifies a logical terminal name associated with a physical terminal. Supply logical terminal names either on the TERMINAL macro LTERM parameter or on a NAME macro that follows a TERMINAL macro. You can associate more than one logical terminal name with a physical terminal.

The MODETBL parameter specifies the name of the VTAM logon mode table *entry* containing the SNA BIND parameters to use when establishing a session for the physical terminal.

You can also supply a mode table entry name on the ISMODE parameter of the Series/1 NETHOST statement used by the Series/1 application program. However, if you specify MODETBL in the IMS/VS system definition, the specified entry name is always used; the Series/1 cannot override the IMS/VS MODETBL log mode entry name with the NETHOST ISMODE parameter.

Use the MODETBL parameter when always using the same mode table entry; use the Series/1 ISMODE parameter to select different entries.

The mode table that contains the entry referenced by MODETBL or ISMODE can be a default table or, as illustrated in the following, you can specify it on the VTAM/NCP LU macro that corresponds to the IMS/VS TERMINAL definition.

1. The VTAM/NCP definition follows:

```
PU1       PU    PUTYPE=2,ADDR=C1,•••                    C
PTERM1    LU    LOCADDR=1,                              C
                PACING=(2,1),                           C
                MODETAB=TABLE1,                         C
                •
                •
                •
```

In the preceding illustration, the VTAM/NCP definition for a mode table called TABLE1 would contain an entry called ENTRY1. Define the BIND mode table parameters for an SLU type P as shown:

```
TABLE1 MODETAB
       MODEENT LOGMODE=ENTRY1,                          C
            FMPROF=X'04',                               C
            TSPROF=X'04',                               C
            PRIPROT=X'B1',                              C
            SECPROT=X'B1',                              C
            COMPROT=X'6080',                            C
            PSERVIC=X'000000000000000000000000'
       MODEEND
```

2. The IMS/VS System definition follows:

```
          TERMINAL NAME=PTERM1,                         C
                   MODETBL=ENTRY1,                      C
                   •
                   •
                   •
          NAME     LTERMA
```

and the Series/1 application ID is as follows:

```
          NETHOST ISAPPID=IMS,                          C
                  ISMODE=ENTRY1,                        C
                  •
                  •
                  •
```

A full discussion of the BIND parameters used for SLU type P appears in the *IMS/VS Programming Guide for Remote SNA Systems.*

The MSGDEL parameter specifies the message types IMS/VS should discard for this terminal. Specifying NONIOPCB prevents the following types of message from being sent to SNA applications:

- Message switches
- Messages inserted by an application program to an alternate program control block (PCB)
- Broadcast messages
- Terminal status messages.

The COMPTn parameters define the separate components for each session. We recommend that you define only one component per session. See "Component Definition" on page D-6 for more information about components.

The OPTIONS parameter specifies communications options associated with this terminal. The following options apply to SNA sessions:

Response mode
Specifies the response mode for this terminal. See "Terminal Response Mode" on page D-6 for more information about response modes.

Message Format Service
Specifies whether MFS is to be used for this terminal. Refer to "Message Format Service" on page D-7 for more information about MFS.

Acknowledgements
Specifies the type of SNA responses used for this terminal. SNA support requires that you specify OPTACK.

Bidding
Specifies whether you use the SNA Bid (BID) command. We recommend using the BID option for SNA terminals.

OPNDST command
Specifies whether you can start this terminal with the /OPNDST command.

The OUTBUF parameter specifies the size of the IMS/VS output buffer. It is also the value passed in the MAXRU field of the BIND command. This value must be within the size limit set for the base system SNA support.

The FPBUF parameter specifies the fast path terminal buffer size. The SNA applications should not send messages on fast path transactions of length larger than FPBUF.

NAME Macro

The NAME macro defines a logical terminal name (LTERM) associated with a physical terminal. LTERM names are specified as positional parameters.

The COMPT parameter specifies the output component associated with the named logical terminal. The ICOMPT parameter specifies the input component associated with the logical terminal. A single component may be capable of both input and output.

Component Definition

IMS/VS considers each physical terminal as being a single program with one or more components defined. As such, each component can have an IMS/VS logical terminal associated with it. The IMS/VS application program can address multiple specific components for each session. IMS/VS performs component selection based on the component indicated in a optional format message header (FMH).

SNA supplies no explicit support for multiple components; therefore, you should define only one component per session. Specify distributed presentation and SNA character string processing only if the application program can handle these options.

Terminal Response Mode

Terminal response mode is a mode of operation that you can define for transactions and terminals attached to IMS/VS. When operating in terminal response mode, a transaction does not end until IMS/VS sends its reply and receives acknowledgment that the reply was received.

You can define terminal response mode as "forced," "negated," or "transaction-dependent." If "forced," every IMS/VS input transaction is in terminal response mode. If "negated," no IMS/VS input transaction is in terminal response mode. If "transaction-dependent," terminal response mode is determined on a transaction-by-transaction basis.

"Forced" terminal response mode removes the need to correlate replies from IMS/VS to the transactions which invoked them. This can reduce the processing required by the SNA application program. For this reason, we recommend "forced" terminal response mode for sessions with SNA applications unless you switch or route messages by alternate program control blocks (PCBs).

The following considerations apply to SNA applications operating in terminal response mode:

- An error preventing transmission of a reply causes a session to wait for data until it abnormally terminates.
- The SNA application cannot perform typical data collection applications since it must receive a reply from the host subsystem before invoking the next transaction.
- The SNA application resends replies remaining on the IMS/VS output queue after session termination when next initiating the session.

Fast Path Feature

Fast Path transactions are single-segment transactions, defined as recoverable by IMS/VS. You must define Fast Path transactions as terminal response mode.

Unlike other IMS/VS output, IMS/VS leaves the Fast Path output message outstanding until the station sends data or an SNA Ready to Receive (RTR) command to dequeue the message. You should consider the RTR command when no input will be generated for an abnormally long time; for example, when the terminal operator plans to leave the terminal. The message or the RTR from the station acknowledges that the preceding IMS/VS output message was received and is recoverable; therefore, IMS/VS may dequeue the message, process any IMS/VS input message, or send any available IMS/VS output message.

Message Format Service

The Message Format Service (MFS) provides data editing capability under the Distributed Presentation Management (DPM) option. SNA provides MFS DPM support for LUs defined as SLU-P and divides responsibility for message formatting between MFS and the SNA application program.

MFS can process data on both input to IMS/VS and output from IMS/VS. Select the availability of MFS on a component basis when defining IMS/VS. SNA provides no explicit support for MFS. The application program must be able to handle the MFS formats and message descriptors when you select MFS.

Refer to "Distributed Presentation Management" for additional information on the use of this option of IMS/VS.

IMS/VS System Definition Considerations

Consider a hardware configuration consisting of a Series/1 system with four terminals, with a requirement to use the terminals for submitting update and inquiry requests to an IMS/VS data base and for receiving the corresponding output data.

You can define the system to IMS/VS in at least two ways:

- As a single physical terminal with four components, each component being capable of input and output and corresponding to a Series/1 terminal.
- As four physical terminals, one per Series/1 terminal, each physical terminal having one component capable of input and output.

In the first approach, the four Series/1 terminals share a single LU-to-LU session, each terminal being addressed by its component identification. Problems with this method can arise because of certain SLU type P protocols. For example, when using terminal response mode, all operations stop between IMS/VS and the physical terminal (in this case the Series/1) from the time IMS/VS receives a transaction until IMS/VS receives acknowledgment that the reply message was received by the Series/1. Therefore, if one of the Series/1 terminals initiates a terminal response mode transaction, the other three Series/1 terminals cannot operate until the transaction completes. This delay may be unacceptable.

The first approach is limited also because a physical terminal is limited to four components. Therefore, you must define additional physical terminals to accommodate more than four Series/1 terminals.

In the second approach, each Series/1 terminal operates with its own LU-to-LU session. Because of this isolation, operation of one terminal does not affect operation of another.

Distributed Presentation Management

Distributed Presentation Management (DPM) is an option of IMS/VS Message Format Service that you can use to format messages for SLU type P devices. This section discusses DPM and compares it to non-DPM formatting.

DPM and the Series/1

With DPM and SLU type P, a device-independent data stream is transmitted between IMS/VS and the Series/1. The Series/1 handles formatting for Series/1 displays and printers. The Series/1 also performs screen formatting. The only data that need be sent by IMS/VS is transaction-variable data. A Series/1 screen formatting function can provide output headers, titles, literals, and default data.

Following is an example of a screen that appears in response to a transaction entered from an IMS/VS terminal:

```
PART=AN960C10          ; DESC=WASHER          ; PROC CODE =74
AREA=2; INV DEPT=80; PRJ=091; DIV=26; PRICE= .000; STK CT DATE=513
CURR REQMTS=  630  ; ON ORDER      15 ; TOTAL STOCK = 695
DISB PLANNED= 1053 ; DISB UNPLANNED=       104 ;  STK CT VARIANCE=0
```

The items to the right of the equal sign (such as the item **AN960C10**) are transaction variable data. The remaining parts of the screen are headers that do not change from execution to execution of the transaction.

You can use at least three different methods to present the above screen:

Method 1 The most primitive method is for the IMS/VS application to construct the whole screen and then transmit it to the terminal in bulk or line-by-line. One disadvantage of this method is that in order to alter the screen to be presented, you must change the IMS/VS application program. Another disadvantage is that even though most of the screen never changes, the whole screen is transmitted every time the transaction executes.

Method 2 A better method is an approach typically used with non-intelligent displays. In this method, you can use MFS to define the constant parts of the screen, including the locations of the fields for variable data. The IMS/VS application program issues only the transaction-variable data, and MFS combines this with its screen definitions to construct the whole screen, which is then transmitted to the terminal. With this method, you do not need to change the IMS/VS application in order to change the screen format. Change the screen format by changing the MFS format definitions. This method still has the disadvantage of transmitting the whole screen for each execution of the transaction.

Method 3 The third method is to use the DPM option of MFS. With this method, only the transaction-variable data need be transmitted to the Advanced Function Controller, that is, the Series/1. A screen formatting function maintains the headers and titles on the Series/1. A Series/1 program integrates the received transaction data with the headers and titles and presents the complete screen to a Series/1 terminal.

Depending on the characteristics of your transactions, you may be able to apply the third method selectively. An installation may have several thousand screens defined. Off-loading this many screen definitions to a Series/1 may be impractical. However, a relatively small percentage of the screens are used most of the time. (For example, 200 out of 2000 screens may be used 85% of the time.) If so, you may be able to

apply off-loading selectively by off-loading the most active screens, leaving DPM with the responsibility for the infrequently used screens.

Selective off-loading is feasible when the transactions for which the host does the formatting are simple output transactions. For transactions such as these, DPM can construct a block of data by combining literals with the message fields output by the IMS/VS application. The block of data is then transmitted to the Series/1, which can treat the data as a screen image and present it to a terminal. Note that device independence may be lost, because DPM constructs the block of data to be a certain size; the Series/1 must be able to present the data to the appropriate device, whose display size may or may not match that of the constructed block of data.

Selective off-loading may not be feasible when the transactions for which the host does the formatting are conversational transactions. For transactions such as these, DPM can construct the output as explained in the preceding paragraph, and the Series/1 can present the data to a terminal. The problem is that the Series/1 has no information to indicate what fields to send to IMS/VS as input. Since DPM does not support literals on input, the Series/1 must send only the fields to be received by the IMS/VS application.

Note: When you use DPM as in the third method explained above, you can use the Message Input Descriptor that is sent in the SLU type P message header to select the appropriate screen definition on the Series/1.

Advantages of DPM

The advantages of using MFS DPM are:

- IMS/VS applications and related MFS formats are device independent.
- The host is off-loaded because the responsibility for screen formatting and device handling transfers to the controller.
- Less data need actually be transmitted, because the controller stores much of the data. Data transmission time is improved, and line contention is reduced.

MFS DPM Compatibility

IMS/VS application programs written for other MFS-supported devices can execute unchanged with SLU type P devices using DPM once you define the MFS device input and output formats as appropriate for the DPM devices.

For example, terminals defined as SLU type 2 are automatically defined to operate with MFS. You can convert IMS/VS applications that communicate with SLU type 2 terminals to SLU type P by altering the IMS/VS system definition, defining new MFS formats, and generating the Series/1 screen definitions.

IMS/VS application program changes may be necessary if the program depends on unique device-dependent features such as premodified fields on a 3270 display. In this case, the program executes unchanged only if the premodified fields presented to the Series/1 return in the input message; this requires that the Series/1 formatting support program properly interpret the attribute bytes of the output message field and handle the indicated device function in a way that satisfies the requirements of the IMS/VS application program.

Refer to the *IMS/VS Message Format Service User's Guide* for more information on this topic.

SLU Type P Without DPM

SLU type P devices do not require use of the DPM option of MFS. You can use three other types of formatting: BASIC, BASIC-SCS1, and MFS-SCS1. (The COMPTn parameter of the TERMINAL macro in the IMS/VS system definition indicates the type used.)

Do not use MFS with BASIC or BASIC-SCS1; use it with MFS-SCS1. BASIC-SCS1 and MFS-SCS1 use SNA SCS data stream format.

If you develop the Series/1 application and the IMS/VS application simultaneously, the BASIC and BASIC-SCS1 formatting options, which specify IMS/VS basic edit, are good alternatives to the use of DPM. With basic edit, you define the format of the data to be transferred, and the two applications are developed to use this format. If the applications agree to transmit only transaction-variable data, basic edit formatting can have the same advantages as DPM formatting.

With basic edit, the Message Input Descriptor name is not available to select the appropriate Series/1 screen definition. This problem is solved by using a data field to convey a screen number or name.

When you use BASIC-SCS1 or MFS-SCS1, deblocking occurs at SCS-defined new line and form feed control characters on input to IMS/VS. Since DPM manipulates fields but is not aware of the contents of the fields, you cannot use DPM to perform deblocking of input to IMS/VS in this manner.

The *IMS/VS Message Format Service User's Guide* and the *IMS/VS Programming Guide for Remote SNA Systems* discuss non-DPM formatting.

Secondary Logical Unit Type P Formats and Protocols

SLU type P formats and protocols define how Series/1 programs accomplish the following:

- Establish sessions with IMS/VS
- Send messages/data to and receive messages/data from IMS/VS
- Terminate sessions with IMS/VS.

Messages/data flowing from IMS/VS to the Series/1 can be:

- Data replies to recoverable or non-recoverable input transactions
- Data replies to IMS/VS commands
- Message switches
- VTAM commands
- VTAM indicators
- IMS/VS system messages
- Broadcast messages
- Null messages (length = 0 bytes) containing EB.

Messages/data flowing to IMS/VS from the Series/1 can be:

- IMS/VS transactions
- IMS/VS commands
- IMS/VS message switches
- VTAM commands
- VTAM indicators
- MFS control requests.

SLU type P protocols define procedures for the exchange of the above types of messages and data. SLU type P formats define the formats of the above types of messages and data.

Selecting SLU Type P Support

Since the Series/1 is programmable, you can write SLU type P support for all of the various protocols and formats. In general, though, a particular application requires only a proper subset of the defined functions; the amount of Series/1 SLU type P support you need to code is minimized by coding to support the subset only.

This section discusses selected portions of SLU type P formats and protocols and some of the options available.

Several system definition options selected on the IMS/VS TERMINAL macro affect SLU type P protocols:

- Whether or not IMS/VS is to assume output component protection
- Whether or not to use terminal response mode, and when.
- Whether or not to perform automatic page deletion for the terminal.
- What type of request acknowledgment protocol to use.
- Whether or not IMS/VS is to send the VTAM BID command.
- What types of messages should IMS/VS allow to be sent to the terminal.
- Whether or not the terminal can send the /OPNDST IMS/VS command.

By limiting how to specify the preceding options, you can minimize the amount of support which you must provide. Both the *IMS/VS Installation Guide* and the *IMS/VS Programming Guide for Remote SNA Systems* discuss each of these items.

Following are some of the other options that affect how much support to give SLU type P function:

- Whether or not to support the Fast Path feature.
- Whether or not to support both recoverable and non-recoverable transactions.
- Whether or not to use MFS paging.
- Whether or not to send IMS/VS message switches.
- Whether or not to send IMS/VS commands.
- Whether or not to support sending of certain VTAM commands, such as QEC.

The *IMS/VS Programming Guide for Remote SNA Systems* discusses all of the preceding options.

Relation to MFS Formatting

When you develop SLU type P support, you must consider the relationship of MFS formatting to SLU type P protocols.

MFS input formatting occurs for data from the Series/1 when you provide a Message Input Descriptor (MID) name with the input message. The MFS Message Output Descriptor (MOD) allows the ability to specify the next MID name to be used to format input as a result of output. Since IMS/VS cannot control the receipt of input and since it is possible that IMS/VS may send a formatted output message while the Series/1 is sending an input message, the Series/1 must ensure that it uses the proper MID name for input formatting. The Series/1 should perform the following steps:

1. Remove the MID name from the received IMS/VS output message header and save it for use on the next input message.
2. Display/print the output message.
3. Get the next operator input.
4. Add the MID name from Step 1 to the input message header.
5. Send the input message to IMS/VS.

When screen formatting has been off-loaded to the Series/1 and DPM is used, the MID name from the received output message header can be used to select the screen definition on the Series/1.

DPM paging also affects SLU type P protocols. You can send fields that make up a presentation page or logical page at one time by specifying OPTIONS=PPAGE or OPTIONS=DPAGE in the device output format control block, The Series/1 must then request additional presentation or logical pages. No paging occurs when OPTIONS=MSG.

Message Resynchronization

SLU type P BIND parameters require that you use SNA transmission subsystem profile 4. This profile specifies that SNA supports the VTAM command Set-and-Test-Sequence-Numbers (STSN).

STSN performs message resynchronization based on the sequence numbers maintained by IMS/VS and the Series/1. You must use sequence numbers correctly

to avoid message loss or duplication. This is particularly important when restarting a session that terminated abnormally.

When sending STSN, IMS/VS uses only the SET and SET AND TEST action codes.

Always use the SET option for the sequence numbers on the flow from the Series/1 to IMS/VS, since IMS/VS knows the sequence number of the last message it received.

SNA can use SET or SET AND TEST for sequence numbers on the flow from IMS/VS to the Series/1. SNA uses SET when no acknowledgment to a recoverable output message is outstanding from the previous session. Use SET AND TEST if IMS/VS sent a recoverable message to the Series/1 but never received the required acknowledgment prior to session termination. The response sent by the Series/1 to the STSN command must indicate whether or not the Series/1 received the message.

Series/1 SNA support sets its sequence number values when it receives a SET or SET AND TEST. When a Series/1 SNA application sends a request, Series/1 SNA support assigns it a sequence number and returns the assigned number to the application. When the Series/1 SNA application sends a response, it must supply the sequence number of the request corresponding to the response. When the Series/1 SNA application receives a request or response, Series/1 SNA support passes the associated sequence number to the application.

The Series/1 application must be able to respond to the SET AND TEST sent by IMS/VS when a recoverable message was sent to the Series/1 but no acknowledgment was received. If the application maintains the sequence number of the last message received on the previous session, an equal compare indicates that the Series/1 received all messages. An unequal compare indicates that the Series/1 did not receive the last message, and retransmission is required.

If the Series/1 application does not retain sequence numbers across sessions, the compare cannot be made, and retransmission is always required.

Advantages of Series/1 as SLU Type P

SLU type P formats and protocols support program-to-program communications. Use of a Series/1 as one or more type P SLUs has the following advantages:

- Since IMS/VS applications communicate with programs in the Series/1 and since Series/1 programs can communicate with a large variety of local and remote input/output devices, a means is provided to integrate many different non-SNA devices into an IMS/VS environment.
- The intelligence that can reside in the Series/1 program can be used to distribute function. The example of off-loading device and screen formatting from IMS/VS was described in "DPM and the Series/1" on page D-8. Further distribution of function is possible. For example, you can develop a Series/1 transaction processing system that can ship only inquires and updates to the IMS/VS data base if they cannot be handled locally.

Example of Series/1 SLU Type P/DPM Configuration

This section presents an example of configuring a Series/1 to communicate with IMS/VS using SLU type P formats and protocols and using the DPM option of MFS.

The elements of the system appear in Figure D-2.

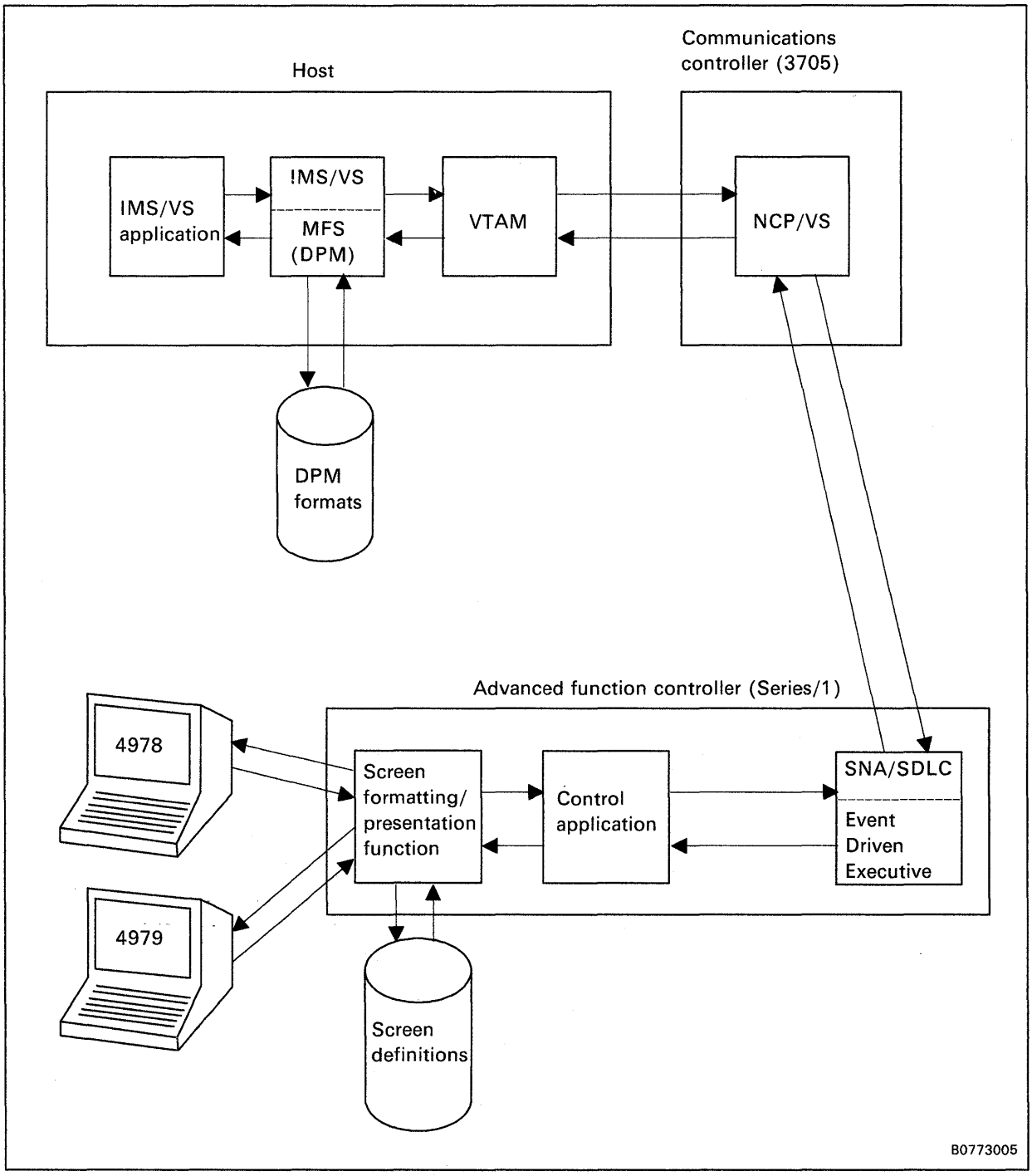


Figure D-2. Series/1 SLU Type P/DPM Configuration

Host Components

In this example, the IMS/VS system definition defines one or more physical terminals as SLU type P terminals and specifies that you must use the DPM option of MFS.

The IMS/VS application program issues transaction-variable data; MFS performs some non-device specific formatting according to the specifications in the DPM formats. This formatting does not construct the whole screen to be presented on the Series/1; transaction-variable data is rearranged, justified in a field, deleted, and so on. Paging to the Series/1 may or may not occur.

The host is off-loaded because the Series/1 handles screen formatting and presentation of the screens to the Series/1 devices. Less data is transmitted because constant parts of the transaction data are not transmitted for each transaction execution.

Series/1 Screen Formatting/Presentation Support

Support must be available to perform screen formatting and presentation of screens to the terminals.

The screen formatting function accepts transaction variable-data output by the IMS/VS application as input. This input is integrated with the stored screen definitions which contain the text of headers and titles to form the complete screen. The screen is then presented to a terminal.

Series/1 Control Application

In the example, there must be a Series/1 application to control the operation of the system. You can write this application in assembler language or a high-level language if the SLU type P support and the screen handling support both provide a CALL interface.

The control application performs the following steps:

1. Calls the SLU type P support to start a session; message resynchronization occurs if necessary.
2. Calls the screen support to present a menu to a Series/1 terminal operator. The operator selects a transaction, and the screen support returns the transaction to the control application.
3. Calls the SLU type P support to send the transaction to IMS/VS.
4. Calls the SLU type P support to receive the transaction data from IMS/VS. The MID name is saved.

5 Calls the screen support, passing it the data received from IMS/VS. The screen support constructs a screen and presents it to the terminal operator. The MID name can be used to identify the appropriate Series/1 screen. If the host formats are defined correctly, no manipulation of the data from IMS/VS is required before passing it to the screen support.

6 Calls the screen support to accept input from the terminal operator.

7 Calls the SLU type P support to send the operator input to IMS/VS. The MID name saved in step 4 is first appended to the message.

8 When all transactions have been completed, calls the SLU type P support to end the session.

CICS/VS Generation

You must describe the Series/1 LUs when defining the CICS/VS system. Each LU used by a Series/1 application in session with CICS/VS corresponds to an entry in the CICS/VS TCT (terminal control table). The CICS/VS macro DFHTCT specifies an entry in the TCT.

Specify the TYPE parameter of the DFHTCT macro as follows:

- If TYPE = INITIAL, specify the following parameter:

  APPLID
  Specifies the VTAM application identification for the CICS/VS system being defined. This is the same as the ISAPPID parameter on the Series/1 NETHOST statement used to initiate a session with CICS/VS.

- If TYPE = TERMINAL, specify the following parameters:

  SESTYPE
  Specify USERPROG for this parameter, indicating a full function or Pipeline LU session.

  TRMTYPE
  Specify 3790 for this parameter.

  TRMSTAT
  Specify TRANSACTION.

Message Formats

The CICS/VS application program and the Series/1 SNA application program determine the conventions of data format. For the full function LU, CICS/VS is not aware of the data format; therefore, it does not provide full Basic Mapping Support (BMS). CICS/VS provides limited BMS support, but SNA provides no explicit support. The SNA application must be able to handle the data formatting required.

Message Recovery/Resynchronization

CICS/VS provides both message recovery and message resynchronization for full function LUs. Message resynchronization occurs when a failed session is restarted, or when an emergency restart is performed for CICS/VS.

The session continues from the last completed logical-unit-of-work (LUW) prior to resynchronization. When a session fails during an incomplete LUW, CICS/VS backs out the effect of the last message it received. CICS/VS retransmits the last message sent from CICS/VS to the Series/1 SNA application when necessary to complete message flow up to the end of the last LUW. See the *CICS/VS System/Application Design Guide* for more information about the LUW concept and suggested transaction restart techniques.

Normally, you should design the CICS/VS application program so that each LUW comprises a CICS/VS input message from the Series/1 SNA application, processing of the message, followed by a committed CICS/VS output message to the Series/1 SNA application. If you do not follow the above restriction, you must design the Series/1 SNA application such that the effect of messages not backed-out by CICS/VS can be either ignored or backed out. For Series/1 SNA sessions established with resynchronization support, SNA enforces the above restriction only to the extent that a reply is required to every message sent to the host subsystem.

Appendix E. Node Definition Statements

This appendix describes host system NCP node generation macros and parameters used to define the Series/1 PU in the SNA network. In addition, the Series/1 configuration statement parameters for a PU corresponding to the NCP node definitions for that PU are described.

NCP Major Node Definition

Figure E-1 shows NCP major node definition operands that should be taken into special consideration when coded to define communications with Series/1 SNA support. Some of the NCP major node definition operands are related to the SNAPU, SNALU, or NETHOST statements on the Series/1; others are related to jumperable options on the Series/1 SDLC attachment. If SNA is using the SDLC support (the SDLC support that is part of the SNA program product), use the SDLC Related SNAPU parameter. If SNA is using shared SDLC support (the SDLC support that is part of EDX Version 6), use the appropriate NDU Shared SDLC Configuration Record parameter in place of the SNAPU parameter. You must code some of the NCP major node definition operands in a specific way to match the characteristics of the Series/1 SNA support.

NCP Node Operand	NCP Node Macro(s)	SDLC Related SNAPU Parameter	Shared SDLC Related SNAPU Parameter	Shared SDLC Related Configuration Record	Shared SDLC Related NDU Parameter	Related SNALU Parameter	Related NETHOST Parameter	Other Related Series/1 Option
ADDR	PU		SECSTAT	SDLC Station	STNx			SDLC attach-ment jumpers
CLOCKING	GROUP LINE							SDLC attach-ment jumpers
DATMODE	PU							
DATRATE	GROUP LINE	RATE		SDLC Line	RATE			
DISCNT	GROUP LINE PU							
MAXDATA	PCCU GROUP LINE PU	BUFSIZ		SDLC Device	IMAX			
MAXOUT	GROUP LINE PU	DCBNO THRESH BUFNO		SDLC Device	RBFN	RECVBUF SENDBUF		
MODETAB DLOGMOD	GROUP LINE PU LU						ISMODE	
NRZI	GROUP LINE	ENCODE		SDLC Line	ENCD			
PACING	GROUP LINE PU LU					RECVBUF		

B0773010

Figure E-1. NCP Major Node Definition Operands

NCP Node Operand	NCP Node Macro(s)	SDLC Related SNPAU Parameter	Shared SDLC Related SNAPU Parameter	Shared SDLC Related Configuration Record	Shared SDLC Related NDU Parameter	Related SNALU Parameter	Related NETHOST Parameter	Other Related Series/1 Option
PUTYPE	GROUP LINE PU							
RETRIES	GROUP LINE PU							
SSCPFM	GROUP LINE PU LU							
ANSTONE ANSWER AUTO CALL DIAL	GROUP LINE	CNCTYPE		SDLC Line	CONN			

B0773011

Figure E-2. NCP Major Node Definition Operands (Continued)

The NCP major node definition operands have the following meanings:

Operand Meaning

ADDR

This specifies the secondary station address for the Series/1 PU being defined. If using SDLC support, the value coded must match the secondary station address jumpered on the Series/1 SDLC attachment. For shared SDLC, the value coded must match the secondary station address jumpered on the card if SECSTAT=FF on the SNAPU statement. For shared SDLC with SECSTAT≠FF, the SECSTAT value must match the ADDR value. In addition, the STNx parameter on the NDU station configuration record must match the SECSTAT specified on the SNAPU statement.

CLOCKNG

This specifies whether the modems or the 3705 and the Series/1 will provide clocking for the line. Whether modem or business machine clocking is specified is installation dependent; however, the value specified should match how the Series/1 SDLC attachment clocking option is jumpered.

DATMODE

This specifies, for a communications controller on a non-switched SDLC link that allows simultaneous transmission in both directions, whether the NCP is to communicate with the distant controller in half-duplex or duplex data mode. The operand DATMODE must be specified as half-duplex (DATMODE=HALF). If excessive retransmission occurs and performance is critical, jumper both the Series/1 and the host modems to half-duplex.

DATRATE

This specifies at which of two data rates a dual-rate modem is to operate. The rate specified is installation dependent; however, the rate selected should agree with the Series/1 PU specification for data rate.

For SDLC, the data rate is specified on the Series/1 by the RATE parameter on the SNAPU configuration statement during Series/1 system generation for the PU.

For shared SDLC, use the RATE parameter on the NDU SDLC Line configuration record.

DISCNT

The YES/NO subparameter specifies whether VTAM should deactivate the PU when the last LU is disconnected. VTAM accepts Request-Discontact commands only when DISCNT=NO is specified. The F/NF subparameter specifies whether VTAM is to indicate "final-use" status when deactivating a PU. When coding this operand, take into account the following characteristics of the Series/1 SNA support:

- The Series/1 specifies "not-last" on all Terminate-Self commands sent.
- The Series/1 sends the Request-Discontact command when you enter either a SNADACT or PUDACT command to deactivate the PU.
- The Series/1 treats all Deactivate-Physical-Unit commands as if they indicate "final-use."

MAXDATA

This specifies the maximum size of path information units (PIUs) that the Series/1 can receive for a PU. The maximum amount of user data the NCP sends to the Series/1 is MAXDATA minus nine bytes.

The Series/1 supports segmenting inbound to the Series/1, but not outbound from the Series/1. The data sent by each NETPUT operation represents a complete request/response unit and must fit in one buffer. When receiving, multiple buffers receive a segmented request/response unit, one PIU per buffer.

For SDLC, the BUFSIZ parameter on the SNAPU configuration statement specifies the buffer allocation size. See the section "SNAPU Configuration Statement" on page 2-5 for information on how to code this parameter.

For shared SDLC, use the IMAX parameter on the NDU SDLC Device configuration record.

MAXOUT

This specifies the maximum number of PIUs the NCP sends to the Series/1 PU before requesting a link-level response.

The number of consecutive PIUs that the Series/1 PU can handle is determined by the number of DCBs and buffers available. Each PIU requires one DCB and one buffer.

For Use with SDLC

The DCBNO parameter of the SNAPU configuration statement specifies the number of DCBs during system generation. If you set DCBNO equal to MAXOUT + 1, enough DCBs are available to receive all PIUs transmitted by the NCP (the +1 is added because the NCP can send an

SDLC control frame after the MAXOUT PIUs). However, you may want to define the Series/1 PU to operate with more or less than MAXOUT + 1 DCBs. See Figure 2-2 on page 2-14 for the advantages and disadvantages of specifying few or many DCBs.

The number of buffers allocated at the Series/1 PU is related to MAXOUT, because each DCB that receives a PIU must point to an available buffer. The number of buffers allocated during PU activation is selected by the BUFNO parameter of the SNAPU configuration statement and the SENDBUF and RECVBUF parameters of the SNALU configuration statement during network generation. The number of buffers allocated (the sum of the value for BUFNO + SENDBUF + RECVBUF) should be equal to or greater than THRESH+1, and THRESH is less than DCBNO. Otherwise, sufficient buffers may not be available to perform message exchange operations.

THRESH keeps the number of available buffers greater than or equal to a defined minimum. When the number of buffers in the pool is greater than the THRESH value, the Series/1 allows the NCP to send messages. When the number of buffers in the pool is less than or equal to THRESH, the Series/1 PU does not allow the NCP to send messages until sufficient buffers accumulate in the pool. See "Threshold Value Considerations (THRESH)" on page 2-12 for a discussion on specifying the value of THRESH.

For Use with Shared SDLC

The MAXOUT on the Group NCP Node macro should be 7. MAXOUT on the PU NCP Node macro should match the RBFN parameter on the NDU SDLC Device configuration record.

MODETAB

MODETAB specifies the name of a logon mode table which the host uses to establish a session with an LU. A log mode entry in MODETAB defines the BIND image to be used for a session.

DLOGMOD

DLOGMOD specifies the name of the logon mode table entry to be used if one is not otherwise provided. For sessions initiated by the Series/1, you can use the mode field of the INIT-SELF request to specify the Series/1 application program's choice of BIND parameters. The ISMODE parameter of the NETHOST statement supplies the mode field. If you code ISMODE, the value specified should be the name of an entry in the table identified by MODETAB. If you do not code ISMODE and you code DLOGMOD, SNA uses the entry identified by DLOGMOD. If you do not code both ISMODE and DLOGMOD, SNA uses the first entry in the table specified by MODETAB.

NRZI

This specifies whether SNA for this PU should use NRZ or NRZI line encoding. The type of encoding specified is installation dependent and should match the type specified at the Series/1 PU.

For SDLC, the ENCODE parameter of the SNAPU configuration statement selects the type of encoding during Series/1 system generation.

For shared SDLC, use the ENCD parameter on the NDU SDLC Line configuration record.

PACING

This specifies, when non-zero, the receive pacing group size, which is the number of normal-flow requests that the NCP can send before awaiting a pacing response. If you are session buffering and you use pacing, the value of the RECVBUF parameter of the SNALU configuration statement needs to be related to the receive pacing group size.

If the number of messages received at one time for the LU is greater than RECVBUF, the session terminates. You can ensure that your session does not terminate by choosing the value of RECVBUF so that it is at least 1 greater than the receive pacing group size. RECVBUF should be at least 1 greater than the receive pacing group size, rather than equal to it, because only normal-flow requests are paced, and an expedited request may flow.

If you are buffer pooling and are using SNA with SDLC support, and you use pacing, the value of DCBNO + SENDBUF + RECVBUF determine the maximum number of messages that can be received at one time for the LU.

PUTYPE

This specifies the PU type. Code PUTYPE=2 for Series/1 PUs.

RETRIES

This specifies the number of attempts the NCP will make to recover from errors occurring during transmission to or from the Series/1 PU. The RETRIES operand has three subparameters. The first specifies the number of retries per retry sequence; the second specifies the time between retry sequences; and the third specifies the number of retry sequences.

In general, code the RETRIES parameter so that more than one retry sequence is performed in the event of a transmission error. This is because retransmissions within a retry sequence occur so quickly that the Series/1 PU may not have enough time to get ready for the retransmission. The minimum allowable specification for the time between retry sequences (one second) should be sufficient for most Series/1 configurations to prepare for the retransmission.

SSCPFM

This specifies whether or not the LU can support character-coded messages in its communications with the SSCP. Code SSCPFM=FSS, because the Series/1 supports only formatted messages from the SSCP.

ANSTONE, ANSWER, AUTO, CALL, DIAL

These operands define how the NCP is to establish the link-level connection. The type of connection indicated by these operands should complement the type of connection specified at the Series/1 PU.

For SDLC, the CNCTYPE parameter of the SNAPU configuration statement can be used to specify a leased, manual answer, auto answer, or manual call type connection. The operator is prompted at network activation time when the connection type is manual call.

For shared SDLC, use the CONN parameter on the NDU SDLC Line configuration record.

Multi-Drop Lines

Be aware of the following conditions when using multi-drop lines.

You should:

- Code the NCP DUPLEX parameter as full.
- Code the NCP DATMODE parameter as half.
- For SDLC only, jumper the Series/1 and host modems or modem eliminator for an 8- to 16-millisecond delay. Shared SDLC requires no such delay.
- For both SDLC and shared SDLC, jumper "data set ready" (DSR) in the modem, and jumper "data terminal ready" (DTR) on the 2090 attachment.

Note: There is no capability for jumping "clear to send" (CTS) on the 2090 attachment.

Sample NCP Major Node Definition for SNA Using SDLC

The sample NCP major node definition that follows is coded to be compatible with SNA support as defined in the Series/1 SNA default system. A discussion of selected aspects of the sample definition follows.

- NCPSER1 is the name of the NCP major node.
- A leased line and a switched line are defined. One PU and one LU are defined per line. Only one line is needed to execute the sample application.
- The leased and switched lines are called LSDLC1 and LSDLC2, respectively. The names of the PU and LU on the leased line are CLSER1 and PGM1, respectively.
- The MAXDATA operand on the leased PU macro specifies 265. This means that the NCP will transmit a maximum of 256 bytes of user data per PIU. In the Series/1 SNA default system, the physical size of buffers allocated is 280 bytes. This means that each Series/1 buffer can handle 259 bytes of user data, and is therefore sufficient to handle any PIU that the NCP will send.
- The MAXOUT operand on the leased PU macro specifies 2. The NCP can, therefore, send up to 2 consecutive PIUs before awaiting a link-level response. The Series/1 SNA default system was defined to have two DCBs and two buffers initially allocated. Because the sample host application sends only one message at a time, the DCBs and buffers will be more than sufficient to handle all NCP transmissions. No retransmissions will be required because of frames not received by the Series/1.

Sample NCP Generation Statements

The following sample illustrates the NCP generation statements for an IBM 3705 in an OS/VS environment.

```
         TITLE 'NCP GEN FOR OS/VS'
*************************************************************************
*************************************************************************
*************************************************************************
*        SUPPORTS SERIES/1 PHYSICAL UNIT                                *
*        THIS GENERATION IS FOR IBM 3705                                *
*************************************************************************
*************************************************************************
*************************************************************************
*        FOLLOWING ARE SOME OF THE NAMES WHICH                          *
*        ARE DEFINED BELOW:                                             *
*                                                                       *
*        NCPSER1  - NAME GIVEN TO THE GENERATED NETWORK                 *
*                   CONTROL PROGRAM LOAD MODULE                         *
*                   (I.E. NAME OF NCP MAJOR NODE)                       *
*                                                                       *
*        CLSER1   - NAME OF PU ON LEASED LINE LSDLC1                    *
*                                                                       *
*        PGM1     - NAME OF LU ON LEASED LINE                           *
*                                                                       *
*************************************************************************
*************************************************************************
*************************************************************************
*        PCCU SPECIFICATIONS - VTAM ONLY                                *
*                                                                       *
*************************************************************************
*************************************************************************
*************************************************************************
NCPSTART PCCU  CUADDR=0B0,        370X CONTROL UNIT ADDRESS            X
               AUTODMP=NO,        PROMPT BEFORE DUMPING NCP            X
               AUTOIPL=NO,        PROMPT BEFORE RELOADING NCP          X
               DUMPDS=VNCPDUMP,   DUMP FILE VTAM DDNAME                X
               INITEST=NO         NO 3705 INITIAL TEST
         TITLE 'BUILD MACRO SPECIFICATIONS'
*************************************************************************
*************************************************************************
*************************************************************************
*                                                                       *
*        BUILD MACRO SPECIFICATIONS                                     *
*                                                                       *
*************************************************************************
*************************************************************************
*************************************************************************
NCPBUILD BUILD MAXSUBA=63,        MUST BE SAME AS IN VTAM STR DEF      X
               LOADLIB=NCPLOAD,   LIBRARY FOR NCP LOAD MODULE          X
               OBJLIB=NCPOBJ,     LIBRARY FOR ASSEMBLER OUTPUTS        X
               QUALIFY=SYS1,      1ST LEVEL QUALIFIER                  X
               TYPSYS=OS,         OS USED FOR STAGE 2                  X
               MEMSIZE=176,       370X STORAGE SIZE IS 176K BYTES      X
               TYPGEN=NCP,        NCP ONLY                             X
               ABEND=YES,         ABEND FACILITY INCLUDED              X
               ANS=YES,           AUTOMATIC NETWORK SHUTDOWN           X
               ASMXREF=YES,       DO ASSEMBLER CROSS-REFERENCE         X
               BFRS=60,           NCP BUFFER SIZE (DEFAULT)            X
```

```
                CHANTYP=(TYPE1),    PRIMARY CHANNEL ADAPTER              X
                ERASE=NO,           DO NOT ERASE BUFFERS (DEFAULT)       X
                DIALTO=60,          WAIT 1 MIN FOR ANSWER                X
                DSABLTO=3.0,        USED WHEN DEACTIVATING LINK          X
                ENABLTO=180,        LARGE ENOUGH FOR DIAL OR LEASED      X
                JOBCARD=MULTI,      JOBCARDS PROVIDED BY NCP GEN         X
                MODEL=3705,                                              X
                NEWNAME=NCPSER1,    NAME OF THIS LOAD MODULE             X
                OLT=YES,            ONLINE TEST AVAILABLE(DEFAULT)       X
                PARTIAL=NO,                                              X
                SLODOWN=12,         SLOWDOWN AT 12% BUFS AVAIL           X
                SUBAREA=9,          SUBAREA ADDRESS = 9                  X
                TRACE=(YES,100)     100 ADDRESS-TRACE ENTRIES
*************************************************************************
*                                                                       *
*        SYSCNTRL OPTIONS REQUIRED BY VTAM                              *
*                                                                       *
*                                                                       *
*************************************************************************
NCPSYSC SYSCNTRL OPTIONS=(MODE,                                          X
                RCNTRL,RCOND,RECMD,RIMM,ENDCALL,                         X
                BHSASSC)
*************************************************************************
*        HOST MACRO SPECIFICATIONS                                      *
*************************************************************************
NCPHOST  HOST   INBFRS=5,           ENOUGH FOR 265 BYTE BATCH PIU        X
                MAXBFRU=3,          VTAM READS 3X116 BYTES               X
                UNITSZ=116,         THREE BUFFERS HOLD BATCH PIU         X
                BFRPAD=28,          OS VTAM 28                           X
                DELAY=.2,           .2 SECOND ATTENTION DELAY            X
                STATMOD=YES,        VTAM LEVEL 2 SUPPORTS STATMOD        X
                TIMEOUT=(120.0)       AUTO SHUT DOWN NO RESP 120 S
        TITLE 'CSB MACRO SPECIFICATIONS'
*************************************************************************
*************************************************************************
*************************************************************************
*        CSB MACRO SPECIFICATIONS                                       *
*                                                                       *
*************************************************************************
NCPCSB   CSB    SPEED=(134,300,600,1200),   BUS MACH CLOCK               X
                MOD=0,              SCANNER ADDRESS 020 TO 03F           X
                TYPE=TYPE2          TYPE 2 COMM SCANNER
        TITLE 'LUPOOL SPECIFICATION'
*************************************************************************
*************************************************************************
*                                                                       *
*      LUPOOL MACRO SPECIFICATION                                       *
*                                                                       *
*************************************************************************
POOL1    LUPOOL  NUMBER=1           ALLOW FOR 1 LU FOR SW PU
        TITLE 'SDLC LEASED LINES'
*************************************************************************
*************************************************************************
*************************************************************************
*                                                                       *
*        SPECIFICATIONS FOR SDLC LEASED LINES                           *
*        GROUP MACRO SPECIFICATIONS                                     *
*                                                                       *
*************************************************************************
```

```
*********************************************************************
*********************************************************************
GROUP1   GROUP  LNCTL=SDLC,       SYNCHRONOUS DATA LINK               X
                DIAL=NO           REQUIRED FOR LEASED LINE
*********************************************************************
*                                                                   *
*        LINE MACRO SPECIFICATION - HALF-DUPLEX, LEASED             *
*                                                                   *
*********************************************************************
LSDLC1   LINE   ADDRESS=00A,      SINGLE LINE INTERFACE ADDRESS       X
                DUPLEX=HALF,      MODEM IS STRAPPED FOR HALF DUP      X
                SPEED=(9600),                                         X
                NRZI=NO,          CHECK WITH MODEM VENDOR             X
                NEWSYNC=NO,       CHECK WITH MODEM VENDOR             X
                CLOCKNG=EXT,      MODEM PROVIDES CLOCKING             X
                POLLED=YES,                                           X
                RETRIES=(5,1,4)   5 RETRIES PER RECOVERY SEQUENCE
*********************************************************************
         SERVICE ORDER=(CLSER1)
*********************************************************************
*********************************************************************
*********************************************************************
*                                                                   *
*        PU MACRO SPECIFICATION FOR SERIES/1 LINK                   *
*                                                                   *
*********************************************************************
*********************************************************************
*********************************************************************
CLSER1   PU     ADDR=C1,          CLUSTER ADDRESS = C1                X
                PUTYPE=2,         SERIES/1 PHYSICAL UNIT TYPE         X
                MAXDATA=265,      MAXIMUM AMOUNT OF DATA              X
                VPACING=(2,1),    PACING TO NCP                       X
                PASSLIM=2,        EQUAL TO MAXOUT                     X
                MAXOUT=2,         MAX PATH INFO UNITS BEFORE RES      X
                ISTATUS=INACTIVE  ACTIVATE VIA OPERATOR
*********************************************************************
*                                                                   *
*        LOGICAL UNIT SPECIFICATIONS                                *
*                                                                   *
*********************************************************************
PGM1     LU     LOCADDR=1,                                           X
                BATCH=NO,         NORMAL DATA PRIORITY                X
                MODETAB=MODETAB1, MODE TABLE                          X
                PACING=(1,1),     PACING TO LU                        X
                ISTATUS=INACTIVE  ACTIVATE VIA OPERATOR
*********************************************************************
*********************************************************************
*                                                                   *
*        SPECIFICATION FOR SDLC DIAL LINE GROUP MACRO               *
*                                                                   *
*********************************************************************
GROUP2   GROUP LNCTL=SDLC,        SYNCHRONOUS DATA LINK               X
                DIAL=YES          REQUIRED FOR DIAL LINE
*********************************************************************
*                                                                   *
*        LINE MACRO SPECIFICATION - half-duplex SWITCHED FOR SERIES/1 *
*                                                                   *
*********************************************************************
LSDLC2   LINE   ADDRESS=00A,      SINGLE LINE INTERFACE               X
```

```
                DUPLEX=HALF,       MODEM IS STRAPPED FOR HALF DUP       X
                SPEED=(1200),                                           X
                NRZI=NO,           UNITS SPECIFIED WITH NRZI            X
                CLOCKNG=INT,       SCANNER PROVIDES CLOCKING            X
                POLLED=YES,                                             X
                RETRIES=(5,1,4),  5 RETRIES FOR RECOVERY SEQUENCE       X
                CALL=INOUT,        INIT CALL FROM NCP/STATION           X
                ANSWER=ON          ALLOW DIAL-IN
*************************************************************************
*                                                                       *
*        PU MACRO SPECIFICATION                                         *
*                                                                       *
*************************************************************************
PU1     PU      PUTYPE=2,          SERIES/1 PHYSICAL UNIT TYPE          X
                MAXLU=1
        GENEND
        END
```

Application Program Major Node Definition

Define a VTAM application program major node by coding one VBUILD statement for the major node and a separate APPL statement for each application program in the major node. The APPL statements define the application program names.

A Series/1 SNA application can identify the VTAM application program with which it wishes to communicate by specifying a value on the ISAPPID parameter of the NETHOST statement that matches a defined VTAM application program name. An alternative is to use a VTAM interpret table to translate the value specified for ISAPPID into the VTAM application program name.

The comments on the MODETAB and DLOGMOD operands in the sample NCP major node definition apply to the APPL operands of the same name.

Sample Application Program Major Node Definition

You can use the sample application program major node definition that follows to define the host sample application:

```
SNAAPPL  VBUILD TYPE=APPL
HOSTPGM1 APPL
```

Note: The Series/1 NETHOST parameter ISAPPID must specify HOSTPGM1 (NETHOST ISAPPID=HOSTPGM1).

Switched Major Node Definition

Figure E-3 shows switched major node definition operands that you should take into special consideration when coding to define communications with Series/1 SNA support.

Switched Node Operand	Switched Node Statement(s)	Related SNAPU Parameter	Shared SDLC Related Configuration Record	Shared SDLC Related NDU Parameter	Related SNALU Parameter	Related NETHOST Parameter	Other Related Series/1 Option
ADDR	PU		SDLC Station	STNx			SDLC attach-ment jumpers
DISCNT	PU						
IDBLK	PU						
IDNUM	PU	STAXID					

Figure E-3 (Part 1 of 2). Switched Major Node Definition Operands

Switched Node Operand	Switched Node Statement(s)	Related SNAPU Parameter	Shared SDLC Related Configuration Record	Shared SDLC Related NDU Parameter	Related SNALU Parameter	Related NETHOST Parameter	Other Related Series/1 Option
MAXDATA	PU	BUFSIZ	SDLC Device	IMAX			
MAXOUT	PU	DCBNO THRESH BUFNO	SDLC Device	RBFN	RECVBUF SENDBUF		
MODETAB DLOGMOD	PU LU					ISMODE	
PACING	PU LU				RECVBUF		
PUTYPE	PU						
SSCPFM	PU LU						

Figure E-3 (Part 2 of 2). Switched Major Node Definition Operands

The switched major node definition operands have the following meanings:

Operand **Description**

IDBLK This specifies the 12-bit block number assigned by IBM to the Series/1. Code IDBLK=021 on each PU statement that corresponds to a Series/1.

IDNUM This specifies the 20-bit identification number assigned to the Series/1 being defined. Specify the value on the Series/1 during system generation on the STAXID parameter of the SNAPU configuration statement.

For a discussion of the remaining operands, see "NCP Major Node Definition" on page E-1.

Sample Switched Major Node Definition Using SDLC

The sample switched major node definition that follows is coded to be compatible with the SNA support as defined in the Series/1 SNA default system for this PU. A discussion of selected aspects of the sample definition follows.

- SWSER1 is the name of the switched PU.
- PGM2 is the name of the switched LU.
- The value specified for IDNUM (X'00000') matches the identification number defined in the Series/1 SNA default system.
- The comments on the MAXDATA operand in the sample NCP major node definition (see "Sample NCP Major Node Definition for SNA Using SDLC" on page E-7) also apply to the MAXDATA operand in the sample switched major node definition.

Sample Switched Major Node Definition Statements

```
*************************************************************************
*************************************************************************
*        SWITCHED SNA MAJOR NODE CONFIGURATION                          *
*                                                                       *
*        FOLLOWING ARE SOME OF THE NAMES WHICH ARE                      *
*        DEFINED:                                                       *
*                                                                       *
*        SWSER1   - NAME OF PU ON SWITCHED LINE LSDLC2                  *
*                                                                       *
*        PGM2     - NAME OF LU ON SWITCHED LINE                         *
*                                                                       *
*************************************************************************
*************************************************************************
         VBUILD TYPE=SWNET,       SWITCHED MAJOR NODE                 X
               SUBAREA=10,        SUBAREA ADDRESS                     X
               MAXNO=1,           NO. OF PHONE NOS. IN SUBAREA        X
               MAXGRP=1           NO. OF GROUP NAMES REF IN SUBA
SWSER1   PU    ISTATUS=INACTIVE,  ACTIVATE VIA OPER                   X
               ADDR=C1,           CONTROLLER ADDRESS=193 DECIMAL      X
               IDBLK=021,         SAME FOR ALL SERIES/1'S IN NET      X
               IDNUM=00000,       UNIQUE FOR EACH SERIES/1 IN NET     X
               MAXDATA=265,       MAXIMUM AMT OF DATA FOR A SERIES/1  X
               PACING=(1,1),      INITIAL SERIES/1 SETTING            X
               VPACING=(2,1),     ALLOW NCP TO BUFFER AREA            X
               MAXPATH=1          NUMBER OF PATHS TO THIS PU
PATHOUT  PATH  DIALNO=2157,       PHONE NO OF THE S/1 FOR DIALOUT     X
               GRPNM=GROUP2,      NAME OF GROUP MACRO IN NCPGEN       X
               PID=1,             UNIQUE FOR EACH PATH PER PU         X
               GID=0              GROUP ID IS 0
PGM2     LU    LOCADDR=1,         LU ADDRESS                          X
               ISTATUS=INACTIVE,  ACTIVATE VIA OPERATOR               X
               BATCH=NO,          NOT BATCH DATA PRIORITY             X
               MODETAB=MODETAB1   MODE TABLE
```

Appendix F. Data Area Descriptions

This appendix provides descriptions of:

- Units of information
- Request/response header
- Transmission header
- Exchange station identification
- SNA buffers.

Units of Information

Following is a description of the basic units of information in an SNA system. Figure F-1 shows the relationships between the units.

- The request/response unit (RU) is the basic unit of information in an SNA system. It can contain application data, commands that control the flow of data through the network, or responses to requests.
- The basic information unit (BIU) is the unit of data and control information that is passed between connection point managers. It consists of a request/response header followed by an RU. (See Figure F-2 on page F-3 for a description of a request/response header.)
- The path information unit (PIU) is the unit of transmission in an SNA system. It consists of a transmission header followed by a BIU. (See Figure F-3 on page F-4 for a description of a transmission header.)
- The basic transmission unit (BTU) consists of one or more PIUs.
- The basic link unit (BLU) is the unit of data and control information transmitted over a data link by data link control. It consists of a BTU preceded by link control header information and followed by link control trailer information.

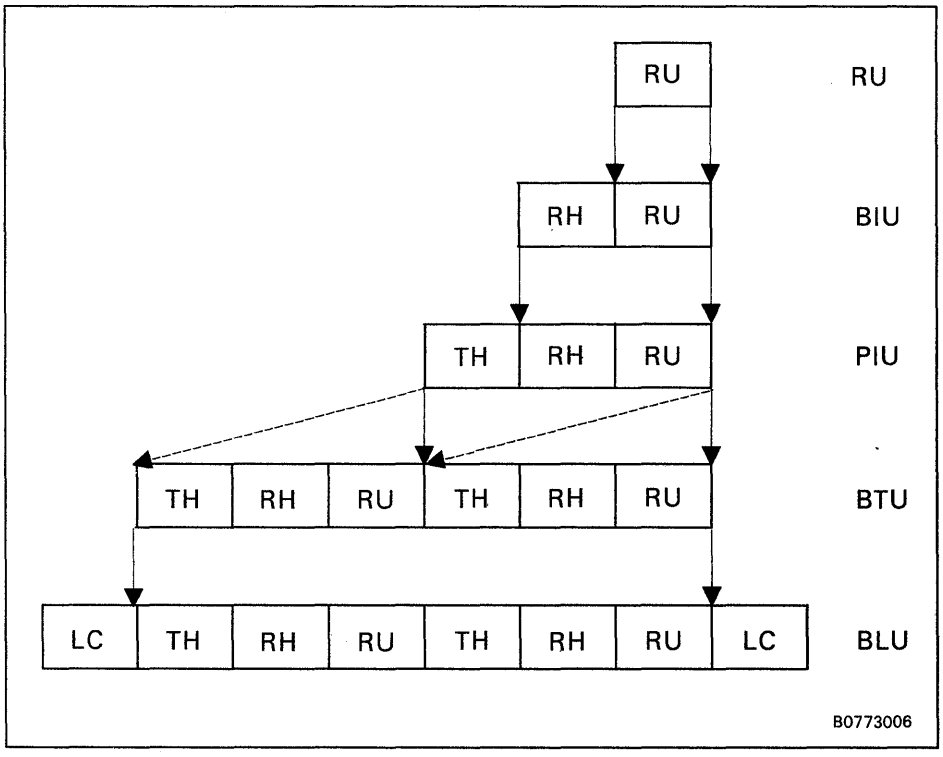
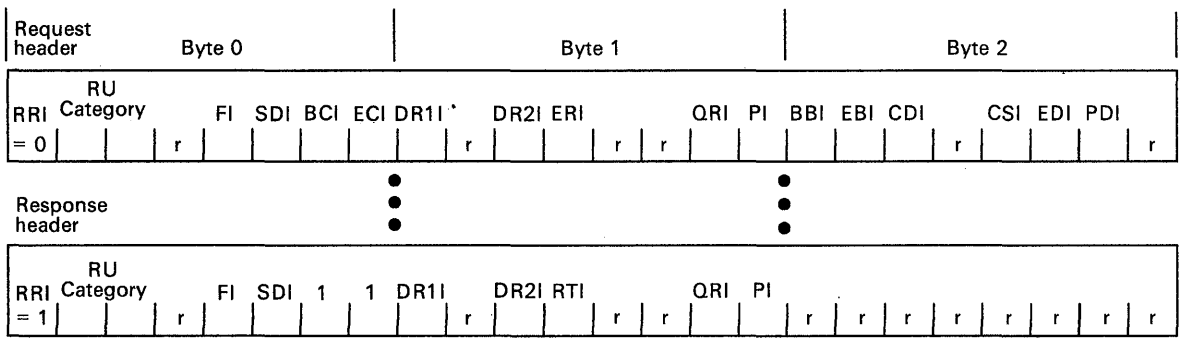


Figure F-1. Basic Units of Information

Request/Response Header

The request/response header is a control field that specifies the type of RU being transmitted (request or response) and control information associated with the RU. Figure F-2 provides a detailed description of the request/response header.



Field	Description	Explanation/Usage
RRI	Request-response indicator	0 = request (RQ); 1 = response (RSP)
RU Category	Request-response unit catagory	00 = FM data (FMD) 01 = network control (NC) 10 = data flow control (DFC) 11 = session control (SC)
FI	Format indicator	0 = no FM header (¬FMH), for LU-LU sessions; or character-coded without an NS header (¬NSH), for network services 1 = FM header (FMH) follows, for LU-LU sessions; or field-formatted with an NS header (NSH), for network services
SDI	Sense data included indicator	0 = not included (¬SD); 1 = included (SD)
BCI	Begin chain indicator	0 = not first in chain (¬BC); 1 = first in chain (BC)
ECI	End chain indicator	0 = not last in chain (¬EC); 1 = last in chain (EC)
DR1I	Definite response 1 indicator	0 = ¬DR1; 1 = DR1
DR2I	Definite response 2 indicator	0 = ¬DR2; 1 = DR2
ERI	Exception response indicator	Used in conjunction with DR1I and DR2I to indicate, in a request, the form of response requested: DR1I, DR2I, ER1 = 000 means no-response requested = 100|010|110 means definite-response requested = 101|011|111 means exception-response requested (001 is reserved)
RTI	Response type indicator	0 = positive (+); 1 = negative (-)
QRI	Queued response indicator	0 = response bypasses TC queues (¬QR) 1 = enqueue response in TC queues (QR)
PI	Pacing indicator	0 = ¬PAC; 1 = PAC
BBI	Begin bracket indicator	0 = ¬BB; 1 = BB
EBI	End bracket indicator	0 = ¬EB; 1 = EB
CDI	Change direction indicator	0 = do not change direction (¬CD); 1 = change direction (CD)
CSI	Code selection indicator	0 = code 0; 1 = code 1
EDI	Enciphered data indicator	0 = RU is not enciphered (¬ED); 1 = RU is enciphered (ED)
PDI	Padded data indicator	0 = RU is not padded (¬PD); 1 = RU is padded (PD)

[r] Reserved

B0773007

Figure F-2. Request/Response Header

Transmission Header

The transmission header is a control field attached to a BIU and used by path control. The sending path control creates it, and the receiving path control interprets it. Figure F-3 shows the transmission header.

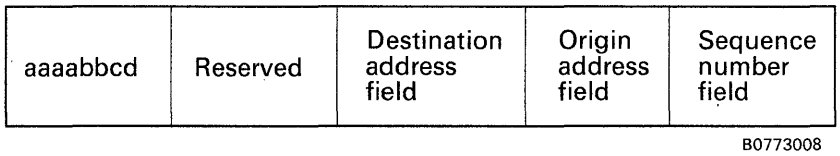


Figure F-3. Transmission Header

where:

aaaa = 0010 = FID type 2
bb = 11 = Mapping field: whole BIU=(BBIU,EBIU)
c = Reserved
d = 0 = Expedited flow indicator: normal
= 1 = Expedited flow indicator: expedited

Exchange Station Identification

Figure F-4 shows the short and long formats of the exchange station identification (XID).

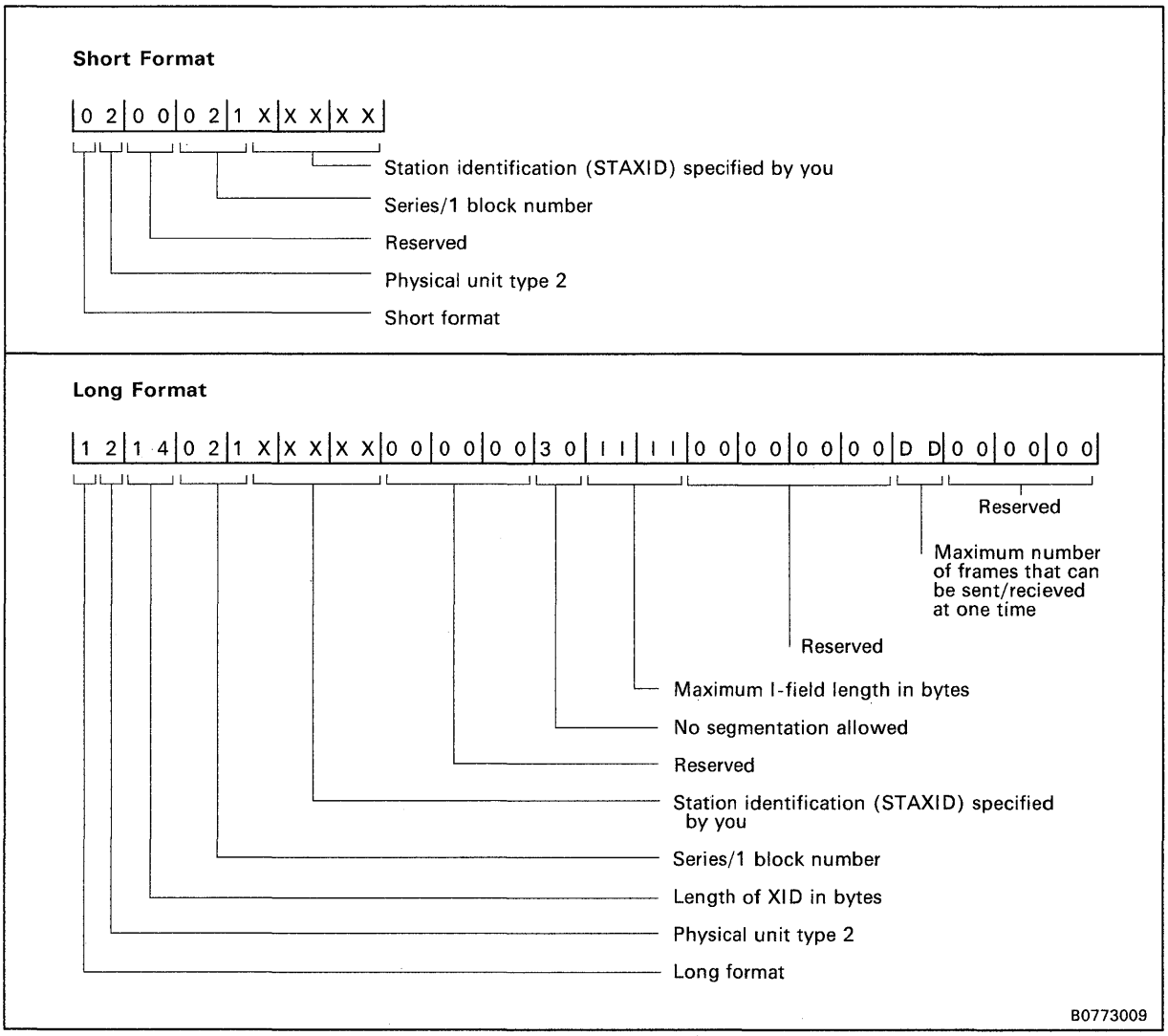


Figure F-4. XID Formats

SNA Buffers

For SNA with SDLC support, SNA and SDLC use the same type of buffer. The number of these buffers in your system is the sum of the following:

- BUFNO value, specified on the SNAPU configuration statement
- SENDBUF value, specified on the SNALU configuration statement
- RECVBUF value, specified on the SNALU configuration statement.

For SNA with extended DLM support, the SNA support does not define the shared SDLC support's buffers. The SNA support only defines the PU and LU buffers. The number of these buffers in your system is the sum of the following:

- BUFNO value, specified on the SNAPU configuration statement
- SENDBUF value, specified on the SNALU configuration statement
- RECVBUF value, specified on the SNALU configuration statement.

When $NETx loads, the buffer count specified by BUFNO determines the number of buffers created and placed into the free buffer pool. When buffer pooling is specified, the buffer counts specified by SENDBUF and RECVBUF determine the number of additional buffers created and added to the free buffer pool. When session buffering is specified, the number of buffers available are determined by each LU. SNA logic uses buffers from the free buffer pool to satisfy all buffer requirements.

Figure F-5 provides a description of the buffer used for SNA.

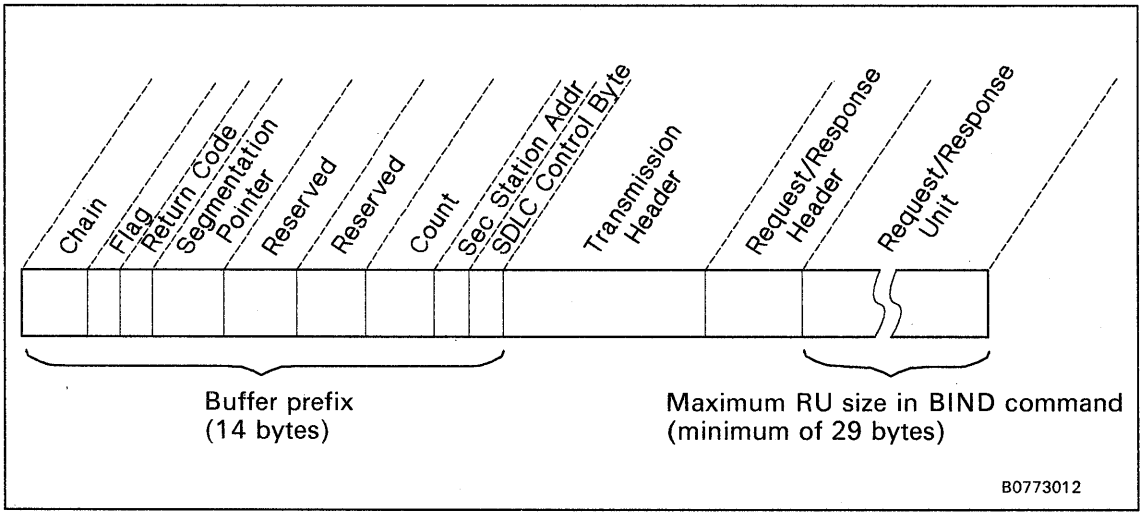


Figure F-5. SNA Buffer

Appendix G. Sense Codes

This appendix describes that subset of the SNA sense codes used by Series/1 SNA support.

REQUEST REJECT

(X'08xx')

This category indicates that the request was delivered to the intended half-session and was understood and supported but not executed:

0801 Resource not available: The LU, PU, or link specified in an RU is not available.

0805 Session limit exceeded: The requested half-session cannot be bound, as one of the NAUs is at its session limit. Applies to INIT, BIND, and CINIT commands.

0809 Mode inconsistency: The requested function cannot be performed in the present state of the receiver.

080B Bracket race error: Loss of contention within the bracket protocol. Arises when bracket initiation/termination by both NAUs is allowed.

0813 Bracket bid reject — no RTR forthcoming: BID (or BB) is received while INB, or while BETB and the first speaker denies permission. RTR is not sent.

0814 Bracket bid reject — RTR forthcoming: BID (or BB) is received while INB, or while BETB and the first speaker denies permission. RTR is sent.

0815 Function active: A request to activate a network element or procedure was received, but the element or procedure was already active.

081B Receiver in transmit mode: A race condition; normal flow request received while the HDX-CONT (or HDX-FF with contention state) FSM state implies contention send (*S,¬R). (Contrast this sense code with X'2004', which signals a protocol violation.)

081C Request not executable: The requested function cannot execute due to a permanent error condition in the receiver.

081D Invalid station/SSCP ID: The station ID or SSCP ID in the request was found to be invalid.

0821 Invalid session parameters: Session parameters are not valid or not supported by a NAU for which the half-session activation is requested.

REQUEST ERROR

(X'10xx')

This category indicates that the RU was delivered to the intended half-session but could not be interpreted or processed. This condition represents a mismatch in half-session capabilities.

1003 Function not supported: The function requested is not supported. The function may have been specified by a formatted request code, a field in an RU, or a control character.

STATE ERROR

(X'20xx')

This category indicates a sequence number error, or an RH or RU which is not allowed for the receiver's current session control or data flow control state.

2001 Sequence number: Sequence number received on normal flow request was not one greater than the last.

2002 Chaining: Error in the sequence of the chain indicator, such as first, middle, first.

2003 Bracket: Error resulting from failure of sender to enforce bracket rules for session. Does not include contention or race resolution indication.

2004 Direction: Error resulting from a normal-flow request being received while HDX-FF FSM state implies protocol violation (S*,¬R). (Contrast this sense code with X'081B', which signals a race condition.)

2006 Data traffic quiesced: An FMD or DFC request received from a half-session which has not sent QUIESCE COMPLETE or SHUTDOWN COMPLETE and has not responded to RELEASE QUIESCE.

PATH ERROR

(X'80xx')

This category indicates that the request could not be delivered to the intended receiver due to a path outage, an invalid sequence of activation requests, or one of the listed PIU errors. (Some PIU errors fall into other categories; for example, sequence number errors are category X'20'.) A path error received while the session is active generally indicates that the path to the session partner was lost.

8004 Unrecognized DAF: An intermediate or boundary PC has no routing information for the DAF, or an end node PC has no LU with indicated DAF(FID1), DAF'(FID2), or LSID(FID3).

8005 No session: No half-session is active in the receiving end node for the indicated OAF-DAF pair, or no BF.SESS.RCV is active for the OAF-DAF pair in a PU.T4 node providing the boundary function. This exception does not apply to BIND, ACTLU, or ACTPU. (This error is listed as a path error since the request cannot be delivered to the intended TC element).

8007 Segmenting error: Mapping field sequencing error, such as first, last, middle; or segmenting not supported and MPF not set to 11. (If segmenting is not supported, a negative response is returned for the first element only, since this contains the RH. Subsequent segments are discarded.)

8008 PU not active: The (SSCP, PU).SEC half-session in the receiving node has not been activated and the request is not ACTPU for this PU. (This error is listed as a path error since the request cannot be delivered to the intended TC element.)

8009 LU not active: A DAF addresses an LU for which the (SSCP, LU).SEC half-session has not been activated and the request is not ACTLU. (This error is listed as a path error since the request cannot be delivered to the intended TC element.)

800F Invalid address combinations: The (DAF',OAF') combination specified an invalid type of session; for example, a PU-LU combination.

Appendix H. Series/1 Network Activation Procedure and Checklist

This appendix describes the Series/1 SNA network activation procedures and a checklist of items to be considered prior to network activation.

Series/1 Network Activation

Series/1 network activation is a procedure that prepares for the connection of the Series/1, by way of a 3705 with an NCP, to an SNA network. The host system network need not be activated before performing Series/1 network activation. However, physical communication cannot begin until the host and the NCP nodes are activated.

The network activation process consists of loading and making resident Series/1 SNA support and enabling the attached SDLC device. The $L $SNA operator command can initiate this process.

If the connection type is manual call after initiation of PU activation, SNA prompts you to call the communications controller. When this message is displayed, dial the telephone number of the 3705 from the telephone on the modem connected to the Series/1 SDLC attachment for that PU. When the telephone at the station is answered, put the modem in data mode. The Series/1 SDLC device for that PU then responds when polled by the NCP.

If the connection type is manual answer or auto-answer, the connection is actually made when the VTAM application starts. The dial-out operation starts at the host when the VTAM application issues the OPNDST. If you coded AUTO= in the NCP, an auto-call unit makes the call from the 3705. Otherwise, SNA prompts the host operator to call the number that appears in the PATH macro in the switched subarea definition.

If the connection type is manual answer, answer the telephone and place the modem in data mode when the telephone rings. Replace the hand set. The SDLC device then responds when polled by the NCP.

If the connection type is auto-answer, the modem is automatically placed in data mode. If the telephone rings, you do not need to manually answer it.

If the connection type is leased, and the Series/1 SNA support for a PU has been loaded, the Series/1 responds when polled by the NCP.

When SNA establishes link-level communications, a message appears on the Series/1 operator station for that PU.

Consider the following items:

- If you have a Series/1 communications indicator panel, you can use it to verify that link-level communications were established. Assuming that you have an indicator panel connected to your SDLC attachment, set the Function/Display switches to 11101000. Lamps 0 – 7 then display the following:

  Lamp 0 Data terminal ready

  Lamp 1 Data set ready

  Lamp 2 Request to send

  Lamp 3 Clear to send

  Lamp 4 Ring indicator

  Lamp 5 Half rate select

  Lamp 6 Transmit mode

  Lamp 7 Receive mode.

  If link-level communications are established, lamps 6 and 7 should blink on and off as the attachment switches between transmit mode and receive mode. Lamps 0 and 1 should both be on. Lamps 2 and 3 should blink on and off, unless the corresponding attachment/modem leads are jumpered to be constantly on.

- If you initiate Series/1 PU activation and the message **NETWORK DEACTIVATED** appears, the Series/1 SDLC attachment may be getting device errors. You can use the system error log for problem determination in this case. Device errors are generally caused by incorrect jumpers or device addresses.

- On a switched line, if the PU deactivates shortly after the activation attempt, verify that the station ID as defined on the STAXID parameter of the SNAPU configuration statement matches the station ID specified on the IDNUM parameter of the PU macro in the switched subarea definitions. Also, verify that the value specified on the IDBLK parameter of the PU macro is **IDBLK = 021**.

- After you enter the PU activation command, if message exchange never occurs, the modems or SDLC attachment may be set up incorrectly. Make sure the modems have the same line speed. For SDLC, check that the modems can handle the type of line encoding (NRZ or NRZI) specified in the Series/1 definition of the ENCODE parameter of the SNAPU configuration statement and the NCP definitions. For SNA with extended DLM. support, check that the modems can handle the type of line encoding (NRZ or NRZI) specified using the Network Definition Utility on the ENCD parameter on the SDLC Line configuration record. For SDLC, verify that the secondary station address jumpered on the Series/1 SDLC card matches the value specified on the ADDR parameter of the PU macro in the switched subarea definition and the NCP definition. For shared SDLC, if the SECSTAT parameter on the SNAPU statement is not "FF", then the SECSTAT value must match the STNx parameter on the NDU SDLC Station configuration record, as well as the value specified on the ADDR parameter of the PU macro in the switched subarea definition and the NCP definition. You can display the secondary station address on the communications indicator panel by setting the Function/Display switches to 01110000.

- PU activation can also fail if the Series/1 SDLC card clocking jumper is installed when it should not be or vice versa. Make sure that the jumper that indicates whether attachment (internal) clocking or modem (external) clocking is correct.
- If the PU activation fails, it may be because a transmission error occurred and the Series/1 could not prepare in time for the retransmission. Refer to the discussion of the RETRIES parameter in "NCP Major Node Definition" on page E-1 if you think this is occurring.

  This problem can also be fixed without recoding RETRIES and doing an NCP generation if your modems have selectable Clear-to-Send delay. Adjusting the modems to increase the time interval between each Request-to-Send and Clear-to-Send gives the Series/1 more time to prepare for retransmission.

Checklist

Following is a list of hardware and software considerations that should be checked by you before activating a network:

1. Modem line speeds defined to the Series/1 and 3705 must match.
2. Modems must both use the same encoding (NRZ or NRZI).
3. If modems do not provide clocking, the SDLC attachment must be jumpered to provide attachment clocking.
4. If you use a switched line, the DTR jumper on the SDLC attachment must be left off.
5. Jumpers other than the DTR jumper — generate answer tone, request to send, and modem delay jumpers — must be set in conjunction with modem options.
6. If you are using X.21 circuit switched network support, make sure that the IBM 2080 attachment card is installed and jumpered.
7. The device address jumpered on the SDLC attachment must match the device address specified on the SNAPU configuration statement (or the device configuration record if using shared SDLC) during Series/1 system generation.
8. The secondary station address definition specified in the ADDR parameter of the VTAM/NCP PU macro must be the same as the secondary station address used for on the Series/1 SDLC attachment.
9. If you are using a switched line, you must code IDBLK=021 on the VTAM/NCP PU macro.
10. The user priority must be lower than that of the SNA inbound message processor.
11. If you are using a switched line, the station identification defined on the IDNUM parameter of the VTAM/NCP PU macro must be the same as the STAXID value specified on the SNAPU configuration statement during Series/1 system generation.
12. The maximum RU size specified in the BIND command (from the System/370 mode table entry or the System/370 application program) cannot be less than the BUFSIZ value specified on the SNAPU configuration statement during Series/1 system generation.
13. The number of LUs defined in VTAM/NCP should be the same as the number defined (number of SNALU configuration statements defined) during Series/1 system generation.
14. The BIND parameters used must be acceptable to the basic BIND check (as specified in Chapter 13, "SNA Protocols").

15 The pacing value defined in the VTAM/NCP definition and mode table must be coordinated with the SENDBUF and RECVBUF values defined on the SNALU configuration statement during Series/1 system generation.

16 If specified, the system services control point identification specified on the SSCPID parameter of the Series/1 NETHOST statement must be the same as that defined to VTAM.

17 You must define the logon mode name specified on the ISMODE parameter of the Series/1 NETHOST statement to VTAM by using the LOGMODE parameter of the MODEENT macro.

18 You must define the application name specified on the ISAPPID parameter of the Series/1 NETHOST statement to VTAM by using the APPLID statement.

19 If the requested System/370 application program has not issued an OPEN macro, you must specify the ISQUEUE=YES parameter of the Series/1 SNA NETHOST statement.

20 You must vary the 3705 NCP program active.

21 If you defined the lines to be inactive, you must vary them active.

22 If you defined the PU to be inactive, you must vary it active.

23 If you defined the LUs to be inactive, you must vary them active.

24 For switched networks, you must activate the switched network subarea.

25 You must vary the VTAM application programs active.

26 You must make sure that no broadcast messages are sent to the Series/1. To ensure this you can default the PU macro on the NCP generation or on the VTAM switched subarea. Code USSTAB=ISTINCDT.

27 You must include timer support in the EDX supervisor, but you do *not* need a timer attachment card.

SDLC Specific Items

1. For SDLC, both modems or modem eliminators must provide delay periods of from 8 to 16 milliseconds for line turnaround. For shared SDLC, this delay is not required.
2. For SDLC, the number of frames sent during one transmission as defined on the MAXOUT parameter of the VTAM/NCP PU macro must be coordinated with the DCBNO, THRESH, and BUFNO values specified on the SNAPU configuration statement and the RECVBUF and SENDBUF values specified on the SNALU configuration statement during Series/1 system generation.
3. For SDLC, link performance using a duplex modem can experience retransmissions when switching from a SEND to a RECEIVE state at the Series/1. The host modem must provide a delay between the RTS signal and the CTS response. A duplex modem keeps the CTS up all the time, which makes it impossible to insert the required 8 to 16 millisecond delay. Strapping a duplex modem RTS/CTS for any delay is ineffective.

   To eliminate the retransmissions, do the following:

   a. Set both modems (host and Series/1) to operate in HALF-DUPLEX, or insure that CTS is not strapped on continuously at the host modem.
   b. Set the "RTS/CTS" delay at the host modem to 8 to 16 milliseconds.
   c. If your NCP PU statement specifies DATMODE=FULL, then change the parameter to DATMODE=HALF and regenerate the NCP.

   Note: The delay of 8 to 16 milliseconds is not required for the shared SDLC support.

Appendix I. SNA Buffer Management

Defining the buffer requirements for an SNA PU is the most difficult step in defining a PU. You must make many decisions concerning:

- Buffer size
- SDLC requirements
- Host requirements
- Buffer pooling versus session buffering
- Total number of buffers.

There are no set rules for defining your buffers for the PU, since each PU has different requirements; however, the following sections contain some guidelines that you can use when defining your buffers.

Buffer Usage

You should have a good understanding of how SNA uses buffers. The SNA facility moves data back and forth between a Series/1 application and a host application, enforcing the rules and protocols defined by the SNA architecture. For a number of reasons, including ease of programming and efficiency, SNA buffers both inbound data and outbound data into storage areas managed internally by SNA. SNA manages these buffers through first-in-first-out queues, except for the SDLC write queue, which is a priority queue.

SDLC Buffer Usage

SDLC has three types of buffers:

- SDLC receive buffers (also called DCB buffers)
- Inbound data buffers
- Outbound data buffers.

To receive data from the network, SDLC sends a chain of receive DCBs to the attachment feature card. Each receive DCB requires one buffer, where the attachment places received data. Each receive DCB can receive one SDLC frame, regardless of the type of SDLC frame. SDLC also has the "short buffer," which is allocated in addition to the buffers you define in the PU definitions. The short buffer is a very small buffer that SDLC uses:

- To complete the DCB chain when not enough buffers are available to fill the entire DCB chain with buffers during a receive operation.
- To send SDLC supervisory frames, such as Receive-Ready (RR) and Unnumbered Acknowledgement (UA).
- To send SDLC error information to the SNA layers in the event of a SDLC permanent error requiring network deactivation.

Before issuing a receive operation, SDLC tries to obtain a buffer for each DCB, with the last DCB using the short buffer. If there are not enough buffers in the free queue to fill all the DCBs, SDLC takes all available buffers and fills as many DCBs

as there are buffers. The short buffer is reserved for SDLC use so that SDLC can always issue a receive operation with at least one buffer. SDLC is the highest priority user of SNA buffers and can always obtain buffers if they are available.

SDLC *never* takes the last buffer from the buffer pool. The last buffer is reserved for sending, so that SNA can always send a REQUEST DISCONTACT to the host as a result of PUDACT or SNADACT.

On the SNAPU statement, you define the minimum number of SDLC Information frames (I-frames) that SNA SDLC must be able to accept per transmission using the THRESH parameter. For example, if you define THRESH=2 and SDLC can receive only one I-frame, Series/1 SDLC cannot accept any I-frames until SNA frees some buffers and it can receive at least two I-frames. THRESH does *not* define the maximum number of I-frames that can be received in one transmission from the host. The maximum number of I-frames allowed is equal to the number of DCBs defined minus 1. The last DCB always accepts the SDLC supervisory frame that host systems send on each transmission. For example, if you define eight DCBs (DCBNO) and THRESH=4, Series/1 SDLC can accept at least four I-frames and cannot accept more than seven in one transmission. For this reason, DCBNO should always be greater than THRESH. The number of DCBs defined should be equal to MAXOUT + 1 defined on the PU statement in the NCP definitions for the Series/1 PU. Since SDLC cannot take the last buffer from the buffer pool, THRESH must always be less than the total number of buffers defined in the system (BUFNO+SENDBUF+RECVBUF).

At the end of each transmit operation, SDLC checks to see if there are enough buffers available in the buffer pool to receive a transmission of at least THRESH number of I-frames. If there are not enough buffers, SDLC sends a receive-not-ready (RNR) supervisory frame to the host. The host upon receiving the RNR continues polling the Series/1 but does not send any I-frames until the Series/1 sends receive-ready (RR). If there are enough buffers in the buffer pool, SDLC sends RR to the host. If SNA runs out of buffers temporarily, SDLC starts sending RNR to the host system, which stops the transmission of all I-frames from the host to the Series/1. When enough buffers return to the buffer pool, the Series/1 begins sending RR, allowing the host to begin sending I-frames again, and both sides eventually recover from the bottleneck.

After receiving data destined for SNA, SDLC removes the buffer from the DCB and places it on the SDLC read queue. After placing the data on the read queue, SDLC posts the SNA inbound message handler. SDLC does this for every message destined for SNA. After processing all SDLC messages, SDLC sets up a transmit operation if a permanent error did not occur.

SNA places all data sent on the SDLC write queue to eventually be sent by SDLC. SNA does not post SDLC to send data. SDLC checks after every received poll for any outbound traffic. So SDLC does not send any data until it receives a poll from the host SDLC station.

After receiving the poll and processing all inbound messages, SDLC checks to see if the host is able to receive I-frames (the host SDLC station may also become overrun and start sending RNRs). If the host can receive I-frames, SDLC checks the SDLC write queue for messages to send. It takes outbound messages from the write queue until the DCB chain is full. Not only does the number of DCBs(DCBNO) determine the maximum number of I-frames that can be received in one transmission, it also determines the number of I-frames that can be sent in one transmission. Note that the messages are not removed from the write queue; they are left on the write queue

even though the DCB chain now points to them also. After placing all the I-frames on the DCB chain, SDLC places an SDLC supervisory command (RR, RNR, and so on) at the end of the chain. Because Series/1 SDLC always sends a supervisory frame at the end of a transmission, the maximum number of I-frames sent per transmission is one less than the total number of DCBs defined. SDLC then issues a transmit operation for the messages.

Outbound messages do not come off the SDLC write queue until the host SDLC station indicates successful receipt of that message. If the host SDLC station does not acknowledge a message and is going to issue another transmit operation (meaning that the host has polled the Series/1), the message is resent. SDLC retries an outbound message the number of times defined by RETRY on the SNAPU statement. If the retry count is exceeded for a message, SDLC flags a permanent error and the PU deactivates. When the host SDLC station acknowledges a message, SNA removes the buffer containing the message from the SDLC write queue and places it on the SDLC read queue with a special indication that it is not inbound data. The SNA inbound message handler takes the buffer, does any processing required, and then frees the buffer. Buffers containing outbound messages are not freed until the host SDLC station acknowledges the message they contain.

SNA Buffer Usage

SNA obtains buffers from the buffer pool when you send messages. If a buffer is not available in the buffer pool, SNA waits for a buffer to be freed for use. If you send a message that is larger than the SNA buffer size, SNA breaks your message into chains. SNA gets enough buffers to send your message and sets the chain indicators in the RH automatically. SNA sends the message as independent chain elements, so that if SNA can get a buffer for the first chain element but not for the second chain element of the message, it sends the first element of the chain and then waits for a buffer to be freed before sending the second element of the chain.

While sending messages, SNA keeps track of how many outstanding send buffers there are for either the LU (if using session buffering) by the SNALU SENDBUF parameter or the PU (if using buffer pooling) by the SNAPU SBUFNO parameter. An outstanding send buffer is a buffer containing a message to be sent by SDLC. For SNA with SDLC support, the buffer is not freed until the host SDLC station acknowledges receipt of the message being sent. For SNA with extended DLM support, the buffer is freed after the data has been copied by shared SDLC. When the buffer is freed, SNA decrements the correct send counts. If you want to send a message and the number of send buffers for either the LU or the PU is at the maximum allowed, SNA waits until a send buffer is freed before sending the message, even if there are buffers available in the free pool. This allows you to restrict the number of buffers available for sending so as to reserve a number of buffers for receive operations. Defining the limit on send buffers does *not* reserve those buffers for send operations. You can use any buffer except the last buffer in the free pool for receive operations. So a send operation may have to wait for a buffer, even though the limit of send buffers outstanding has not been reached.

For received data, the SNA inbound message handler removes messages from the SDLC read queue and places them on the appropriate session queues. For multi-segmented messages, the message handler does not place the message on a session queue until all the segments for the message are received. The SNA inbound message handler queues all the segments on a PU segmentation queue until the full message is built and then gives the data to the session. After placing the message on the appropriate session queue, the SNA inbound message handler posts required

events (internal SNA receive events and the attention event for the session if you supplied it on the NETINIT instruction). The SNA inbound message handler then waits for the next message to be received from the SDLC support.

After placing the data on the session queues, you can now receive the data by issuing a NETGET instruction. SNA handles some data (positive responses to NETPUT messages that were sent with VERIFY = YES) and does not require NETGET. SNA cannot process the messages that it handles internally until the SNA inbound message handler places the message on the correct session queue and posts the SNA internal receive event. The data passes through the various layers of SNA architecture and is finally copied to your data areas.

Note that SNA cannot process the data until it receives all segments for that message. If you receive a message that has more segments than there are free buffers currently available, some buffers must be freed before SNA can process the message. This is a very common hang condition in SNA where not all the segments have been received for a message and no more buffers will be freed, because either there weren't enough buffers defined in the network or the applications are not issuing NETGETs for sessions that currently have messages available. If a buffer is freed, SNA can recover and receive the entire message. After SNA processes the message (if it is an internal message) or your application copies all the data, the buffer containing that message is freed.

As with send operations, there can be a limit on the number of messages queued on a session waiting to be received by the applications. You define this limit only if the PU uses session buffering. If using buffer pooling, you do not define a limit, and SNA can perform receive operations until there are no more buffers available. For session buffering, you set a limit on the number of outstanding receive buffers allowed for each LU with the RECVBUF parameter on the SNALU statement. A receive buffer is a buffer containing data to be received by a session. Both user data and internal SNA messages are counted as receive buffers. Only LU-LU messages are counted, and responses to expedited messages are not counted. When your application or SNA completely receives the data, the buffer is freed and is no longer counted as a receive buffer for that session.

Note: Remember that each buffer required to hold a received message is counted as one outstanding receive buffer. If a message consists of three segments, the SNA layers treat it as one complete message, but the message requires three receive buffers.

If a session exceeds its receive buffer limit, the session terminates. If a three-segment message is received and the session has a receive buffer limit of two (RECVBUF = 2), SNA terminates the session.

Buffer Pooling Versus Session Buffering

Buffer pooling and session buffering are the two buffer management schemes that SNA provides. You choose the scheme to use with the BUFPOOL parameter on the SNAPU statement. Buffer pooling shares the data buffers among the PU and LUs equally, while session buffering reserves data buffers for the PU and each of the LUs. Therefore, buffer pooling allows the most efficient use of the available buffers, but session buffering allows independence between separate sessions.

Descriptions of Buffer Pooling and Session Buffering

Buffer Pooling

In buffer pooling, all buffers are shared and are placed in a single free pool. Whenever a LU or PU needs one, it obtains a buffer from the free pool. The number of outstanding send buffers is restricted on a PU basis by the SBUFNO parameter or the sum of all SNALU SENDBUF parameter values.

Note: Unlike BUFNO, SENDBUF, and RECVBUF, SBUFNO does *not* allocate additional buffers.

Therefore, the number of buffers containing messages waiting to be sent from the PU or any LU cannot exceed the value defined. For example, if SBUFNO=4 and you defined three LUs with LU #1 having two messages waiting to be sent, LU #2 having one message waiting to be sent, and the PU having an outstanding send message, LU #3 cannot send any data until one of the previous messages is sent and acknowledged. If the send count is exceeded, the task issuing the message (either your application or SNA) waits for a message to be sent and the buffer to be freed.

You can use any buffer for receiving data (except the last buffer in the pool). So, if there are 10 buffers defined in the system and four are defined as send buffers, the support can use nine of these buffers to receive data and four of these buffers for sending data when they are not used for receiving data.

For SDLC, if there are no buffers available to receive data, SDLC sends RNR until there are sufficient free buffers available to receive inbound messages.

For SNA with extended DLM support, if there are no buffers available to receive data, then the extended DLM support in SNA cannot ask the shared SDLC support for inbound messages until there is a free buffer available.

Session Buffering

With session buffering, all buffers are in a single free pool, but each LU and the PU have limits on the number of buffers available for use. The PU defines a set of buffers (using the BUFNO parameter on the SNAPU statement) that are not available to the LUs and are used primarily for SDLC, messages destined for the PU, and all expedited messages. Each LU defines a finite set of send and receive buffers using the SENDBUF and RECVBUF parameters on the SNALU statement. The total number of buffers used for sending or receiving messages for that LU cannot exceed the defined limits.

As in buffer pooling, SDLC or the extended DLM support in SNA can use any buffer to receive data, except for the last buffer in the free pool. To SDLC or the extended DLM support in SNA, there is no difference between session buffering and buffer pooling. But when the SNA inbound message handler determines that the message is destined for a specific LU, it checks to see how many buffers are currently being used by that LU for receiving data. If the maximum receive count for that LU (defined by RECVBUF) is exceeded, the SNA inbound message handler discards the received message and terminates the session.

Session buffering limits the total number of send buffers used in the system, similar to the method used by buffer pooling. But unlike buffer pooling (which only limits the total number of send buffers used by the PU and the LUs), session buffering defines send buffer limits for each LU within the PU. If a LU tries to send a message and its send buffer limit has already been reached, the operation waits until a send buffer from that LU is freed, even if there are available buffers in the free pool. The session does not terminate, as it does if the receive buffer count is exceeded. Because the SENDBUF parameters on the SNALU statements define the number of allowed send buffers, you must set SBUFNO to zero for session buffering.

Buffer Pooling and Session Buffering Trade-Offs

There are advantages and disadvantages to using either session buffering or buffer pooling. The main advantage of buffer pooling over session buffering is that it usually requires far fewer buffers to be defined, thereby decreasing the size of the SNA storage requirements. But session buffering allows session independence, whereas buffer pooling allows a single session with high buffer demands to affect the performance of the other sessions.

For example, if a PU has four LUs defined, three of which are used for CF 3270 passthru terminals and the other is used as an RJE station, the performance of the terminals can be directly affected by the RJE LU. If RJE starts receiving a large print job requiring many or all of the buffers allocated with buffer pooling, the performance of the passthru terminals can be noticeably slower, because no buffers are available for the terminal messages. With session buffering, the performance of the 3270 terminals are not affected by the RJE LU, because each LU has a set of buffers reserved for its exclusive use. Even in this example, the performance of the terminals are not affected as much if the RJE LU uses SNA pacing to control the number of messages being sent or received. See "Pacing" on page I-16.

It is not easy to choose between buffer pooling and session buffering. Both methods have advantages and disadvantages for any type of environment. Usually, buffer pooling is the better of the two methods because of the savings in storage and the efficient use of SNA resources. Also, you can modify situations which at first appear to require session buffering to use buffer pooling by the intelligent use of SNA pacing and application tuning.

The advantages of buffer pooling are:

- Lower storage requirements
- More efficient use of SNA resources
- Use of the full power of SNA buffer management
- Non-termination of sessions when overrun by received data.

The advantages of session buffering are:

- Session independence (can be obtained in buffer pooling by the use of pacing)
- More control over the flow of data through SNA.

A disadvantage of buffer pooling is that the performance of one LU may be affected by the amount of data transmission from another LU.

The disadvantages of session buffering include:

- Most of the time some of the SNA buffers are not used and storage is wasted
- The session can be terminated if overrun by inbound messages.

Buffer Pooling Definitions

The relevant parameters for buffer pooling are BUFPOOL, BUFNO, and SBUFNO on the SNAPU statement and SENDBUF and RECVBUF on the SNALU statement. For buffer pooling, BUFPOOL must be YES. The total number of buffers defined for the PU is equal to the sum of BUFNO, SENDBUF for each LU, and RECVBUF for each LU. This is the total number of buffers defined in the PU. The total number of outstanding send buffers allowed is equal to either SBUFNO or the sum of SENDBUF for all the SNALU statements. If SBUFNO is equal to zero, the sum of the SENDBUF values must not be zero. If SBUFNO is not equal to zero, both SENDBUF and RECVBUF for each of the LUs must be equal to zero.

For SDLC (the SDLC support that is part of the SNA program product), all considerations for SDLC buffer usage (DCBNO, THRESH, and so on) still apply for buffer pooling. (For example, DCBNO > THRESH and the total number of SDLC buffers equals BUFNO + SENDBUF + RECVBUF > THRESH + 1.)

For shared SDLC, SDLC buffer usage no longer impacts SNA buffer pooling. The SDLC buffers are in a separate pool determined by the RBFN and TBFN parameters on the NDU SDLC Device configuration record. The SNA buffer pool now only services the PU's and LU's requirements.

Sample Definitions for Buffer Pooling for SNA with SDLC

The following illustration is a sample definition of a PU with two LUs that uses buffer pooling. There are a total of eight buffers in the buffer pool. There can be a maximum of four send operations outstanding at one time. This PU can receive and transmit a maximum of three I-frames per transmission (DCBNO - 1). On receive there must be at least three buffers (THRESH + 1) in the free pool (remember, SDLC cannot take the last buffer in the free pool) or else SDLC sends RNR.

```
NET1        SNAPU •                                          C
                  •                                          C
                  •                                          C
                  DCBNO=4,                                   C
                  THRESH=2,                                  C
                  BUFNO=8,                                   C
                  BUFPOOL=YES,                               C
                  SBUFNO=4
            SNALU LU=1,                                      C
                  •                                          C
                  •                                          C
                  •                                          C
                  SENDBUF=0,                                 C
                  RECVBUF=0,                                 C
                  END=NO
            SNALU LU=2,                                      C
                  •                                          C
                  •                                          C
                  •                                          C
                  SENDBUF=0,                                 C
                  RECVBUF=0,                                 C
                  END=YES
```

The following illustration is a sample definition of a PU with two LUs that uses buffer pooling. There are a total of twelve buffers in the buffer pool. There can be a maximum of four send operations outstanding at one time. This PU can receive and transmit a maximum of three I-frames per transmission (DCBNO - 1). On receive there must be at least three buffers (THRESH + 1) in the free pool (remember, SDLC cannot take the last buffer in the free pool) or else SDLC sends RNR.

```
NET1        SNAPU •                                  C
                  •                                  C
                  •                                  C
                  DCBNO=4,                           C
                  THRESH=2,                          C
                  BUFNO=4,                           C
                  BUFPOOL=YES,                       C
                  SBUFNO=0
            SNALU LU=1,                              C
                  •                                  C
                  •                                  C
                  •                                  C
                  SENDBUF=2,                         C
                  RECVBUF=2,                         C
                  END=NO
            SNALU LU=2,                              C
                  •                                  C
                  •                                  C
                  •                                  C
                  SENDBUF=2,                         C
                  RECVBUF=2,                         C
                  END=YES
```

Sample Definitions for Buffer Pooling for SNA with Shared SDLC

The following illustration is a sample definition of a PU with two LUs that uses buffer pooling. There are a total of eight buffers in the buffer pool. There can be a maximum of four send operations outstanding at one time.

The shared SDLC support requires the MAXDCB=16 on the EXIODEV statement for the SDLC attachment card. This allows the shared SDLC support to chain 8 DCBs for receive and 8 DCBs for transmit operations. The shared SDLC support can the receive and transmit a maximum of 7 I-frames per transmission.

The shared SDLC support has separate receive and transmit buffers. The number of receive buffers is specified by RBFN. The number of transmit buffers is specified by TBFN. If the extended DLM support has no buffer available to insure a READ request to the shared SDLC support for data, then the shared SDLC support will continue to receive the data from the host until it exhausts all of its receive buffers. When the extended DLM support obtains a buffer for a READ request, then the shared SDLC will pass the received data to the extended DLM support for processing. Whenever the shared SDLC support exhausts all of its receive buffers, then it issues RNR to the host.

```
NET1        SNAPU •                                          C
                  •                                          C
                  •                                          C
                  BUFNO=8,                                   C
                  BUFPOOL=YES,                               C
                  SBUFNO=4
            SNALU LU=1,                                      C
                  •                                          C
                  •                                          C
                  •                                          C
                  SENDBUF=0,                                 C
                  RECVBUF=0,                                 C
                  END=NO
            SNALU LU=2,                                      C
                  •                                          C
                  •                                          C
                  •                                          C
                  SENDBUF=0,                                 C
                  RECVBUF=0,                                 C
                  END=YES
```

The following illustration is a sample definition of a PU with two LUs that uses buffer pooling. There are a total of 12 buffers in in the buffer pool. There can be a maximum of four send operations outstanding at one time.

```
NET1        SNAPU •                                          C
                  •                                          C
                  •                                          C
                  BUFNO=4,                                   C
                  BUFPOOL=YES,                               C
                  SBUFNO=0
            SNALU LU=1,                                      C
                  •                                          C
                  •                                          C
                  •                                          C
                  SENDBUF=2,                                 C
                  RECVBUF=2,                                 C
                  END=NO
            SNALU LU=2,                                      C
                  •                                          C
                  •                                          C
                  •                                          C
                  SENDBUF=2,                                 C
                  RECVBUF=2,                                 C
                  END=YES
```

Session Buffering Definitions

The relevant parameters for session buffering are BUFPOOL, BUFNO, and SBUFNO on the SNAPU statement and SENDBUF and RECVBUF on the SNALU statement. For session buffering, BUFPOOL must be NO and SBUFNO must be zero. The total number of buffers defined for the PU is equal to the sum of BUFNO, SENDBUF for each LU, and RECVBUF for each LU. This is the total number of buffers defined in the PU. Each LU uses the RECVBUF parameter to define the maximum number of outstanding receive buffers allowed for the LU and the SENDBUF parameter to define the maximum number of outstanding send buffers allowed for the LU. Because SNA exchanges messages internally with the host, you must never specify SENDBUF and RECVBUF as zero when using session buffering; you should specify them as at least 2.

For SDLC, all considerations for SDLC buffer usage (DCBNO, THRESH, and so on) still apply for buffer pooling. (For example, DCBNO > THRESH and the number of SDLC buffers equals BUFNO + SENDBUF + RECVBUF > THRESH + 1.)

For shared SDLC, the SDLC receive and transmit buffers are in a separate pool allocated by the shared SDLC support. Therefore, SDLC buffer usage has no impact on SNA buffer usage.

Because the session terminates if the number of allowed receive buffers for a LU is exceeded, session buffering usually requires the host application to use outbound pacing for the session. The only time this is not required is if you are absolutely sure that the host application is not going to overload the secondary LU with messages (for example, it uses definite response for every message sent).

Also because of the consequences of overflowing the receive buffer count, you must take great care to consider segmentation when defining the number of receive buffers.

Sample Definitions for Session Buffering

The following illustration is a sample definition of a PU with two LUs that uses session buffering. There are a total of twelve buffers defined in the free buffer pool for the PU. Both LUs can have two outstanding send operations. Both LUs also require that no more than two receive buffers be queued up for each LU at any time.

Figure I-1 illustrates the SNA network definition for use with the SDLC support in SNA. Figure I-2 on page I-13 illustrates the SNA network definition for use with the shared SDLC support.

```
NET1        SNAPU •                                          C
                  •                                          C
                  •                                          C
                  DCBNO=4,                                   C
                  THRESH=2,                                  C
                  BUFNO=4,                                   C
                  BUFPOOL=NO,                                C
                  SBUFNO=0
            SNALU LU=1,                                      C
                  •                                          C
                  •                                          C
                  •                                          C
                  SENDBUF=2,                                 C
                  RECVBUF=2,                                 C
                  END=NO
            SNALU LU=2,                                      C
                  •                                          C
                  •                                          C
                  •                                          C
                  SENDBUF=2,                                 C
                  RECVBUF=2,                                 C
                  END=YES
```

Figure I-1. SNA Network with SDLC Support

```
NET1        SNAPU •                                          C
                •                                            C
                •                                            C
                BUFNO=4,                                     C
                BUFPOOL=NO,                                  C
                SBUFNO=0
            SNALU LU=1,                                      C
                •                                            C
                •                                            C
                •                                            C
                SENDBUF=2,                                   C
                RECVBUF=2,                                   C
                END=NO
            SNALU LU=2,                                      C
                •                                            C
                •                                            C
                •                                            C
                SENDBUF=2,                                   C
                RECVBUF=2,                                   C
                END=YES
```

Figure I-2. SNA Network with Shared SDLC Support

The following illustration is a sample definition of a PU with two LUs that uses session buffering. There are a total of twenty buffers defined in the free buffer pool for the PU. Both LUs can have two outstanding send operations. LU #1 limits the number of allowed receive buffers to four, while LU #2 allows up to eight buffers to be queued waiting for either SNA or the application to receive the data contained within them.

Figure I-3 on page I-14 illustrates the SNA network definition for use with the SDLC support. Figure I-4 on page I-14 illustrates the SNA network definition for use with the shared SDLC support.

```
NET1        SNAPU •                                          C
                 •                                           C
                 •                                           C
                 DCBNO=4,                                    C
                 THRESH=2,                                   C
                 BUFNO=4,                                    C
                 BUFPOOL=NO,                                 C
                 SBUFNO=0
            SNALU LU=1,                                      C
                 •                                           C
                 •                                           C
                 •                                           C
                 SENDBUF=2,                                  C
                 RECVBUF=4,                                  C
                 END=NO
            SNALU LU=2,                                      C
                 •                                           C
                 •                                           C
                 •                                           C
                 SENDBUF=2,                                  C
                 RECVBUF=8,                                  C
                 END=YES
```

Figure I-3. SNA Network with SDLC. Support

```
NET1        SNAPU •                                          C
                 •                                           C
                 •                                           C
                 BUFNO=4,                                    C
                 BUFPOOL=NO,                                 C
                 SBUFNO=0
            SNALU LU=1,                                      C
                 •                                           C
                 •                                           C
                 •                                           C
                 SENDBUF=2,                                  C
                 RECVBUF=4,                                  C
                 END=NO
            SNALU LU=2,                                      C
                 •                                           C
                 •                                           C
                 •                                           C
                 SENDBUF=2,                                  C
                 RECVBUF=8,                                  C
                 END=YES
```

Figure I-4. SNA Network with Shared SDLC Support

Segmentation and Buffer Size

Segmentation occurs when the upper layers of SNA send a message down to path control, which is larger than the maximum size of the SDLC I-frame for the PU. The message must be split into parts so that it can be sent through SDLC. The corresponding path control layer on the other side of the SDLC link reassembles the segments into the original message. SNA implements segmentation to allow session independence from the SDLC characteristics of the PU on which the session operates. For example, a session may specify that the maximum message allowed between session partners is 512 bytes (specified in the BIND command for the session). You can establish the session using LUs on PU #1 or PU #2. The maximum size of a SDLC I-frame for PU #1 is 521 bytes (512 bytes of request unit (RU) plus six bytes for the transmission header (TH) and three bytes for the request header (RH). All messages sent when the session is running on PU #1 require only one segment to be sent. For PU #2, the maximum I-frame size is 265 bytes, so a message larger than 256 bytes requires two segments. The applications through the data flow appear exactly the same no matter which PU the session is on.

SNA supports inbound segmentation (messages received from the host can be segmented). SDLC or the extended DLM support in SNA receives each segment into a receive buffer and passes the buffer to the SNA inbound message handler. Then the SNA inbound message handler reassembles the segments into complete messages and passes them to the corresponding session queues. The message handler cannot place a message on a session queue until it receives all segments for that message.

For messages being sent from the Series/1 that are larger than the maximum I-frame size for the PU, SNA separates the message into chains, not segments. SNA does this so that maximum throughput can be achieved using the limited amount of buffers available in SNA. If path control segments a message, it cannot send any other messages from any other LU until the other SDLC station successfully receives all segments for that message. But if SNA splits a message into chains, other messages can be sent between the chain elements if they come from a higher priority source.

Although it is only one message, each inbound segment requires a receive buffer to be received by SDLC and SNA. Therefore, SNA counts each inbound segment of a message against the outstanding receive count of a session.

The host defines the maximum size of an SDLC I-frame for the PU with the MAXDATA parameter. This specifies, in bytes, the maximum allowed size of an I-frame sent by the host to the secondary SDLC station. On the Series/1, you define the size of your buffers using the BUFSIZ parameter on the SNAPU statement. This defines the maximum RU size that the PU can receive or send. The value for BUFSIZ must be at least equal to the value of MAXDATA − 9, and can be larger. If MAXDATA = 265, you should specify BUFSIZ as 256. SNA adds the size of TH and RH to BUFSIZ, along with the size of the SNA header (14 bytes) to calculate the actual size of the buffer within SNA. You can specify BUFSIZ to be larger than MAXDATA − 9. This allows you to send messages that are larger than MAXDATA − 9, which is allowed by NCP; however, on receive operations that extra storage is wasted.

For shared SDLC, the BUFSIZ parameter on the SNAPU statement must be 9 bytes less than the IMAX field on the SDLC device configuration record.

Pacing

You may have to use pacing for LU-LU sessions to control the flow of data to and from the Series/1. SNA supports both inbound and outbound pacing, which is transparent to your applications. Pacing is useful for either session buffering or buffer pooling, although it is usually far more important with session buffering.

Pacing limits the number of messages that you can send to a session partner. Primarily, it prevents one session partner from overloading the other session partner. Pacing groups your messages together in what is called a "pacing group." The session partner cannot send any of the messages within a pacing group until the other session says it can accept all the messages within the group.

Expedited messages are not affected by pacing and do not apply to pacing protocols.

A session indicates that it is ready to receive the next pacing group by sending a pacing response. A pacing response is sent independent of any other responses that flow for the session and indicate only that the LU is willing to accept the next pacing group. Each pacing group has one message that carries the pacing indicator (PI) in its RH. The position within the pacing group varies, but there is only one per group. The pacing response is a response to this message and may be sent anytime after receipt of the message with PI set. The pacing request, when sent, refers to the messages in the next pacing group along with the remaining messages in the current pacing group. For example, if the size of a pacing group is four messages and the pacing request is the second message within the group, the pacing request asks if the session partner can accept six total messages (four in the next pacing group and messages three and four in the current pacing group).

Define pacing in the BIND command for the session. The BIND defines both inbound and outbound pacing group size and the number of the message within the group that have the PI set. It is important to note that pacing relates to SNA messages, not SDLC I-frames. Even though a message may be split into multiple segments, it is still counted as only one message within the pacing group.

SNA handles inbound pacing requests (messages with PI set) after your application copies the message for that request and the buffer is about to be placed in the free pool as follows:

For session buffering

If the LU receiving the pacing request has enough receive buffers available to accept the next pacing group or if it has the maximum amount of receive buffers allowed for the LU (no outstanding receive buffers available for the LU), SNA sends the pacing response.

Otherwise, SNA sets an indicator showing that a pacing response is due. Every time a receive buffer is freed for that LU, the above conditions are checked and if satisfied, SNA sends the pacing response. Note that SNA does not send the pacing response until a receive buffer is freed, so SNA does not send pacing responses until after either SNA or the application have received the message from the host. When calculating whether the LU has enough receive buffers available, SNA takes into account the remaining buffers to be received in the current pacing group, along with the buffers in the next pacing group.

For buffer pooling

SNA sends the pacing response immediately.

SNA also supports outbound pacing by sending pacing requests and waiting for pacing responses from the host. SNA sets the pacing indicator on the first message of each new pacing group. As SNA sends each message for a pacing group, it checks the pacing counter (the size of the pacing group). If the pacing counter is zero and the pacing response for the next pacing group was not received, the message cannot be sent. In this case the message is queued onto the session outbound pacing queue and processing continues. NETPUT still completes successfully if the message was put onto the pacing queue and not actually sent to SDLC. If the pacing counter is not zero, it is decremented and the message is placed on the SDLC write queue. The SNA inbound message handler handles pacing responses. When SNA receives a pacing response for a session, it adds the size of the next pacing group to the pacing counter. For example, if the pacing group size is 3 and the host immediately responded to the pacing request, SNA sets the pacing counter to 5 (2 from the current pacing group and 3 for the next pacing group). If there are messages currently on the pacing queue waiting to be sent, the SNA inbound message handler sends the messages until the queue is empty or the pacing counter is exhausted.

Pacing with Session Buffering

Pacing at least inbound messages is very critical for most sessions whose PUs use session buffering. This ensures that the receive count is not overrun and the session does not terminate because of high data traffic. If the applications do not have guarantees that they will send only a certain amount of messages before waiting for a response from the Series/1, you must use inbound pacing which has a direct effect on the number of receive buffers defined for a LU (RECVBUF). Because SNA does not send a pacing response until it has enough receive buffers available and SNA remembers the current number of messages available in the current pacing window, the minimum requirement is that there must be enough receive buffers to receive a full pacing window. The formula for RECVBUF for a LU using session buffering is:

```
int((size of a msg)/(BUFSIZ))*(m) = RECVBUF (minimum)

where : size of a msg => size of the message being received
        BUFSIZ        => size allowed per buffer
        m             => pacing group size
```

This ensures that the LU will not terminate its session because the receive buffer count was exceeded. The above formula is the minimum value allowed unless the sessions have specific rules. Because pacing also limits the number of messages that can be received, there is also a maximum practical value for RECVBUF. The formula is:

```
int((size of a msg)/(BUFSIZ))*(2*m-n) = RECVBUF (maximum)

where : size of a msg => size of the message being received
        BUFSIZ        => size allowed per buffer
        m             => pacing group size
        n             => number of msg with PI in pacing group
```

The LU can have more receive buffers defined for the LU, but that LU never uses them. Some configurations may specify RECVBUF higher in order to allocate more buffers for the PU and SDLC, but you should allocate those by the SNAPU BUFNO parameter.

You can use outbound pacing with session buffering, which may be of some use for some host applications. But from the Series/1 perspective it has no effect whatsoever on buffer usage. The SENDBUF parameter of the SNALU statement actually controls send operations. But since pacing limits the number of messages that you can send at any one time, it imposes a practical maximum value for SENDBUF. The formula is:

```
        2*m-1 = SENDBUF (maximum)

where:  m            => pacing group size
```

Because SNA does not segment outbound messages, you do not need to calculate segmentation adjustments in the formula. Also, because SNA always sends the pacing indicator on the first message of a pacing group, n = 1 always. You can define SENDBUF as some smaller value. If defined less than the maximum value, SNA buffer management controls when messages are sent. If you specify the maximum, the LU never waits for a buffer because it has exceeded its send count. It may still have to wait if buffers are tied up for receive operations. As with RECVBUF, you can specify SENDBUF larger, but the extra buffers are queued only on the pacing queue, waiting for pacing responses to be received if the pacing counter is exceeded.

Pacing with Buffer Pooling

You can use pacing with buffer pooling to limit the number of outstanding receive buffers allocated for a particular LU. Because SNA does not send the pacing response until the message containing the pacing indicator is received, the largest amount of receive buffers tied up for a particular LU is equal to (m-n) + m, where *m* is the size of the pacing group, and *n* is the number of the message within the group that carries the pacing indicator. Remember to take into account the number of segments needed per message. The actual formula is:

```
int((size of a msg)/(BUFSIZ))*(2*m-n) = outstanding receive buffers

where : size of a msg => size of the message being received
        BUFSIZ        => size allowed per buffer
        m             => pacing group size
        n             => number of msg with PI in pacing group
```

You can use outbound pacing with buffer pooling; it may have some use for some host applications. But from the Series/1 perspective, it has no affect whatsoever on buffer usage. SBUFNO actually controls send operations.

Summary

This appendix presents guidelines, not rules, that you can use when defining the buffer requirements for EDX SNA. You must not break some rules (for example, THRESH must always be less than the total number of buffers defined in the PU), but you should treat most as guidelines. Every environment is different and you should evaluate your resources and objectives (for example, storage versus response time) when defining your buffers. Usually, you must tune an SNA network and change the buffer definitions at least once before the system performs to your satisfaction. You should use this appendix to evaluate your buffer needs and make educated choices for your buffer definitions.

Index

Special Characters

A

B

C

D

E

J

L

M

N

O

P

Q

T

U

V

W

X

Numerics

IBM Series/1 Event Driven Executive Systems Network Architecture Version 2 and Remote Job Entry Guide

Order No. SC34-0773-1

READER'S COMMENT FORM

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

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

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

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.)

SC34-0773-1
Printed in U.S.A.

Reader's Comment Form

Cut or Fold Along Line

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
Information Development, Department 28B
5414 (Internal Zip)
P.O. Box 1328
Boca Raton, Florida 33429-9960

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

Program Number
5719-XX9, 5719-SX2

File Number
S1-30

SC34-0773-1

Printed in U.S.A.