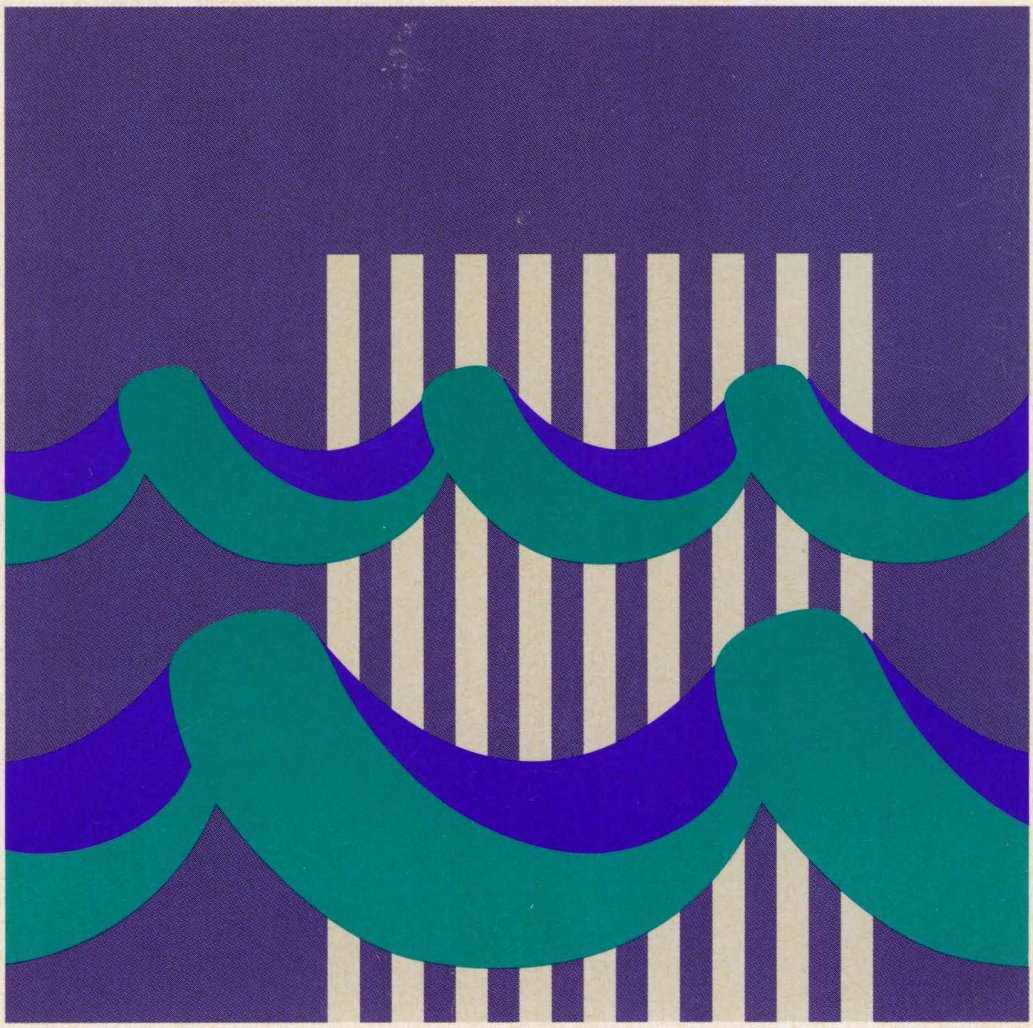
LY43-0031-01

Network Control Program
System Support Programs

Customization Guide

NCP Version 7 Release 2
SSP Version 4 Release 2



LY43-0031-01

Network Control Program
System Support Programs

Customization Guide

NCP Version 7 Release 2
SSP Version 4 Release 2

Note

Before using this document, read the general information under "Notices" on page xi.

Second Edition (October 1994)

This major revision replaces LY43-0031-00. This licensed document applies to the following IBM licensed programs:

- Advanced Communications Function for Network Control Program Version 7 (program number 5648-063).
- Advanced Communications Function for System Support Programs Version 4 for MVS (program number 5655-041), for VM (program number 5654-009) Release 1, and for VSE (program number 5686-064) Release 1.

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch office serving your locality.

A form for your comments is provided at the back of this document. If the form has been removed, you may address comments to:

IBM Corporation
Department E15
P.O. Box 12195
Research Triangle Park, North Carolina 27709
U.S.A.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Notices

Any reference to an IBM licensed program in this licensed document does not imply that IBM intends to make it available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
International Business Machines Corporation
500 Columbus Avenue
Thornwood, New York 10594
United States of America

The licensed programs described in this document and all licensed material available for them are provided by IBM under terms of the IBM Customer Agreement.

This document is not intended for production use and is furnished as is without any warranty of any kind, and all warranties are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

Programming Interface Information

This book is intended to help the customer customize Advanced Communications Function for Network Control Program (NCP) and System Support Programs (SSP). This book primarily documents Product-Sensitive Programming Interface and Associated Guidance Information provided by NCP.

Product-Sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-Sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

However, this book also documents General-Use Programming Interface and Associated Guidance Information.

General-Use programming interfaces allow the customer to write programs that obtain the services of SSP.

General-Use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

```
┌──────────── General-Use Programming Interface ────────────┐
```

General-Use Programming Interface and Associated Guidance Information...

```
└──────────── End of General-Use Programming Interface ─────┘
```

Trademarks

The following terms, denoted by an asterisk (*) at their first occurrence in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

BookManager	MVS/XA	VTAM
IBM	NetView	VSE/ESA
Library Reader	VM/ESA	
MVS/ESA		

About This Book

This book provides information to help you customize Advanced Communications Function for Network Control Program (NCP) and System Support Programs (SSP).

Customizing NCP means modifying it to enhance support for certain stations or to provide support for stations not currently supported by the IBM-supplied programs. Customization can also include writing programmed Systems Network Architecture (SNA) resources that reside in the communication controller.

Customizing SSP means modifying it to control the generation process for both NCP and Emulation Program (EP). You can create generation applications that process the NCP or EP generation definition as part of the SSP generation process. This enables you to define network resources not provided by the IBM-supplied definition functions.

Who Should Use This Book

This book is for programmers who want to modify or enhance an NCP or the generation process for NCP and EP. You must have a detailed knowledge of the function and operation of NCP. You must also be familiar with the basic concepts of data communication and SNA.

How to Use This Book

Refer to this book to familiarize yourself with NCP and SSP customization and for help in writing routines to customize NCP line control and programmed resources. Use *NCP and SSP Customization Reference* for information about the NCP and SSP customization macros provided with the program.

Before you begin to create generation applications, read Chapter 4 to learn how to write and use these applications. As you code your generation applications, refer to the alphabetical listing of NCP/EP definition facility (NDF) internal utilities in *NCP and SSP Customization Reference* for detailed information on those utilities.

Terms Used in This Book

The following descriptions explain how terms are used in the NCP, SSP, and EP library.

"MVS," "VM," and "VSE"

The term *MVS* means the MVS/XA* and MVS/ESA* systems. The term *VM* means the VM/ESA* systems in the CMS environment. The term *VSE* means the VSE/SP, VSE/ESA*, and VSE/Advanced Function operating systems. If information is applicable to only one system, the specific system name is used.

"Port" and "Channel" with LPDA

In discussions concerning link problem determination aid (LPDA) for multiport and data-multiplex mode (DMPX) modems, the terms *port* and *channel* are synonymous. Although *port* is the more commonly used term, *channel* can be used in sections describing LPDA.

"IBM Special Products or User-Written Code"

This book sometimes refers to *IBM special products or user-written code*. This phrase means IBM* special products such as Network Terminal Option (NTO), Network Routing Facility (NRF), and X.25 NCP Packet Switching Interface (NPSI), or user-written code.

IBM 3745 Communication Controller Model Numbers

In this book, the term *IBM 3745 Communication Controller* refers to all IBM 3745 models. When particular models are discussed, the appropriate model numbers are specified. Model numbers include IBM 3745-130, 3745-150, 3745-160, 3745-170, 3745-17A, 3745-210, 3745-21A, 3745-310, 3745-31A, 3745-410, 3745-41A, 3745-610, and 3745-61A.

"Ethernet-Type LAN"

The term *Ethernet-type LAN* means a local area network (LAN) that uses either the Ethernet Version 2 or IEEE 802.3 protocol.

"NCP V7R2"

In this book, unless otherwise specified, the term *NCP V7R2* refers to NCP Version 7 Release 2 with or without the optional NCP feature for 3746 Model 900 connectivity subsystem support. To use this feature, you must have the 3746 Model 900 installed in your controller.

How Numbers Are Written

This book shows numbers over 9999 in metric style, which means that a space is used instead of a comma to separate groups of three digits. For example, the number ten thousand five hundred fifty-two is written 10 552. However, keyword values, for example, SALIMIT=65535, do not use a blank.

Symbols Used in This Book

Figure 1 shows the networking symbols used in the illustrations that appear throughout this book.

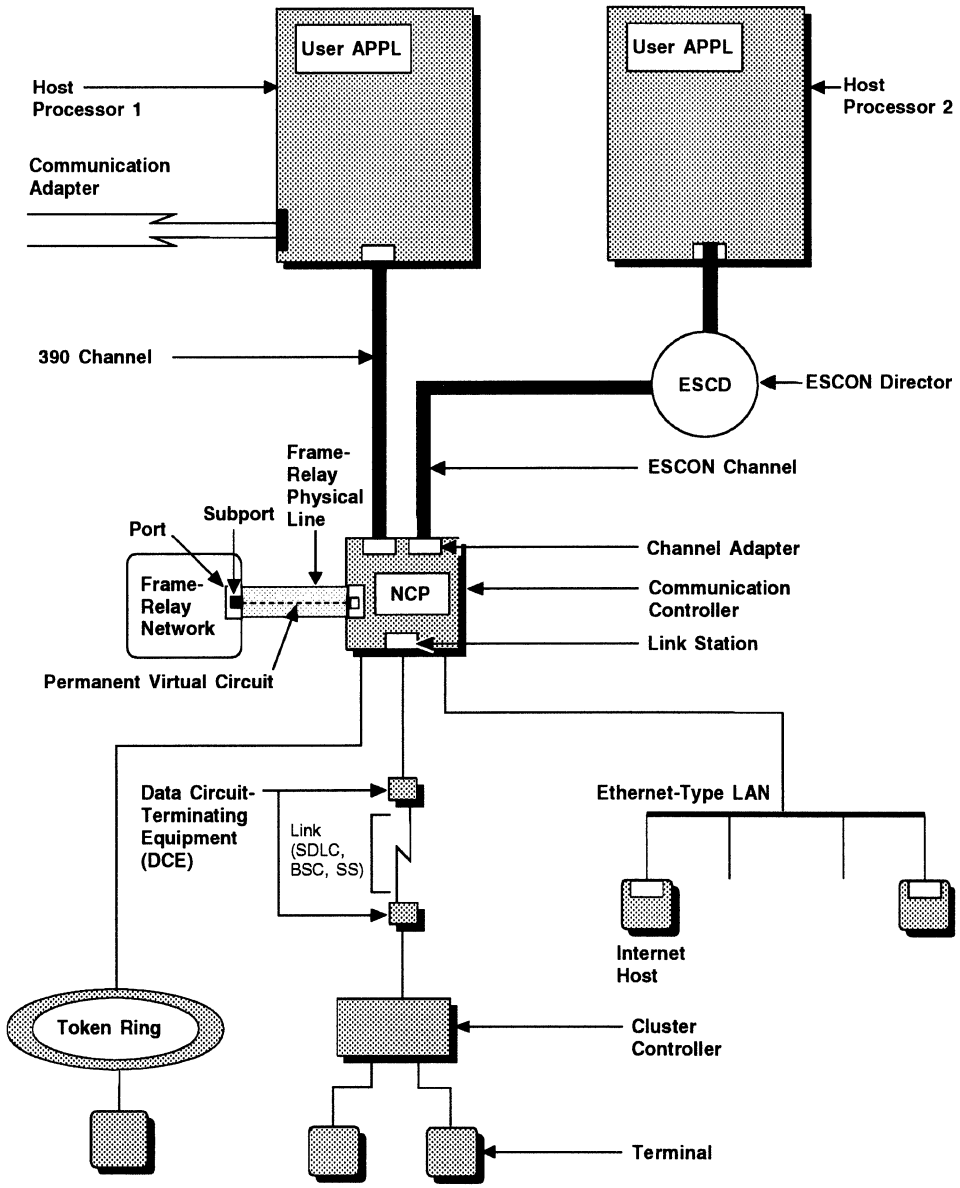


Figure 1. Symbols Used in Illustrations

What Is New in This Book

This edition contains no new editorial or technical changes.

Where to Find More Information

A good place to start any task regarding NCP, SSP, or EP is *NCP V7R2, SSP V4R2, and EP R12 Library Directory*. This directory introduces the enhancements for the current release and shows where these enhancements are described in the NCP library. It gives you an overview of NCP, SSP, and EP and directs you to information on a variety of tasks related to these programs. When you are using the book online, you can use *hypertext links*[1] to move directly from task and enhancement descriptions to the appropriate chapters of other books in the library.

Information for NCP Tasks

The books in the NCP, SSP, and EP library are listed here according to task, along with closely related books and tools you may find helpful. See "Bibliography" on page 157 for a brief summary of each book in the NCP, SSP, and EP library and listings of related publications.

Table 1 (Page 1 of 2). Sources of Information by Task

Order No.	Title	Hardcopy	Softcopy
Planning			
SC31-7122	*Planning for NetView, NCP, and VTAM*	■	■
SC31-7123	*Planning for Integrated Networks*	■	■
SX75-0092	*Planning Aids: Pre-Installation Planning Checklist for NetView, NCP, and VTAM*	■	
SC31-6259	*NCP V7R2, SSP V4R2, and EP R12 Library Directory*	■	■
Installation and Resource Definition			
SC31-6221	*NCP, SSP, and EP Generation and Loading Guide*	■	■
SC31-6258	*NCP V7R2 Migration Guide*	■	■
SC31-6223	*NCP, SSP, and EP Resource Definition Guide*	■	■
SC31-6224	*NCP, SSP, and EP Resource Definition Reference*	■	■
Customization			
LY43-0031	*NCP and SSP Customization Guide*	■	
LY43-0032	*NCP and SSP Customization Reference*	■	

D Available on diskette for the IBM OS/2 environment.

[1] A *hypertext link* is a pointer from a location in an online book to another location in the same book or another book. By selecting highlighted information, such as a message number, you can move quickly to related information and, if desired, back again.

Table 1 (Page 2 of 2). Sources of Information by Task

Order No.	Title	Hardcopy	Softcopy
Operation			
SC31-6222	*NCP, SSP, and EP Messages and Codes*	■	■
N/A	*Online Message Facility*		D
Diagnosis			
LY43-0033	*NCP, SSP, and EP Diagnosis Guide*	■	
LY43-0037	*NCP, SSP, and EP Trace Analysis Handbook*	■	
LY43-0029	*NCP and EP Reference*	■	
LY43-0030	*NCP and EP Reference Summary and Data Areas*	■	
LK2T-1999	*NCP, SSP, and EP Diagnosis Aid*		D
Monitoring and Tuning			
SC31-6247	*NTune User's Guide*	■	■
LY43-0035	*NTuneNCP Reference*	■	

D Available on diskette for the IBM OS/2 environment.

Those publications available as softcopy books have cross-document search and hypertext links for speedy, online information retrieval. These softcopy books are grouped together on an electronic bookshelf and are part of the *IBM Networking Systems Softcopy Collection Kit* on compact disc read-only memory (CD-ROM).

You can view and search softcopy books by using BookManager* READ products or by using the IBM Library Reader* product included on CD-ROM. For more information on CD-ROMs and softcopy books, see *IBM Online Libraries: Softcopy Collection Kit User's Guide* and BookManager READ documentation.

Part 1. NCP Customization

Chapter 1. Writing Customized NCP Routines

Before you can customize Advanced Communications Function for Network Control Program (NCP), you must have a thorough understanding of Systems Network Architecture (SNA) and NCP. Refer to the bibliography for a list of books containing SNA information. *NCP and EP Reference* includes a detailed description of NCP operation and concepts.

NCP routines are written in 3705 Assembler Language, an extension of 370 Assembler Language. Refer to *3704/3705 Communications Controller Assembler Language* for more information. *NCP and SSP Customization Guide* assumes that you are familiar with 3705 Assembler Language.

This chapter explains the NCP operating environment, how to use the IBM-supplied NCP macros available for customization, how to write structured NCP code, how to use routine types common to both line control and programmed resources, and how to generate your customized NCP load module.

Understanding the NCP Operating Environment

The IBM communication controller is a transmission control unit. Its processing is controlled by programs such as NCP that reside in its storage. The communication controller, together with NCP, performs many of the functions previously assigned to the central processing unit or access method.

NCP receives data from the host processor and transmits it to terminals, clusters, or other NCPs. NCP also receives data from the terminals, clusters, or other NCPs and sends it to the host processor. NCP processes the data as it passes through the controller. In controlling the flow of data, NCP must interact with the controller hardware. NCP interacts with the communication scanner to control line communication and with the channel adapter to control channel communication.

The following sections describe levels of programming, tasks in NCP, the NCP supervisor, and interval timer routines that affect you as you add IBM special products or user-written code to your generation definition.

Levels of Programming

The communication controller is an interrupt-driven control unit. The NCP that controls its operation is made up of smaller programs or levels. Interrupts can be caused by the channel, communication lines, or the program itself.

The controller has five program levels. Program level 1 has the highest priority and program level 5, referred to as the background level, has the lowest priority. Because level 5 has the lowest priority, its code runs when levels 1 through 4 are not executing. See *IBM 3745 Communication Controller Introduction* for a complete description of the five levels of the controller and the interrupt facility. See *NCP and EP Reference* for more information on how NCP uses program levels.

Tasks in NCP

A *task* in NCP is a portion of code and a queue of data upon which the code operates. These tasks are defined only in level 5. If one portion of code operates upon two or more separate queues of data, this portion of code is handled as two or more separate tasks by the task dispatcher. The background level (level 5) of NCP is made up of several tasks that work together to schedule line activity and process messages. Only one level 5 task is active in NCP at any one time.

A task is defined during NCP generation or initialization when a queue control block (QCB) is assembled and linked to a unit of code. As queues become activated, their associated tasks are scheduled and initiated by the task dispatcher. *Input queues* (input to a task) are activated by the enqueuing of data to the queue. Enqueuing can be done by the communication interrupt control program (level 3) when a block is received from the communication lines, or by path control (SDLC) or the system router (BSC and start-stop) when a PIU is received over the channel, or the enqueuing can be done in the background level as one task passes control to another task. *Pseudo-input queues* (recording a stimulus for the task, but providing no data as input to the task) are activated by triggering the task at the occurrence of a stimulus, such as a panel display request.

Figure 2 on page 7 shows a task schematic within NCP. The code is modularly structured with a distinct division of function. The task sequencer within each task interfaces directly with the supervisor by receiving control when the task is dispatched and returning control when task execution ends. It also controls communication with other tasks by the enqueuing of data or by the registering of stimuli. The task sequencer controls the execution of the task by calling the appropriate subroutines in the correct sequence. A subroutine executes a logically concise process and returns.

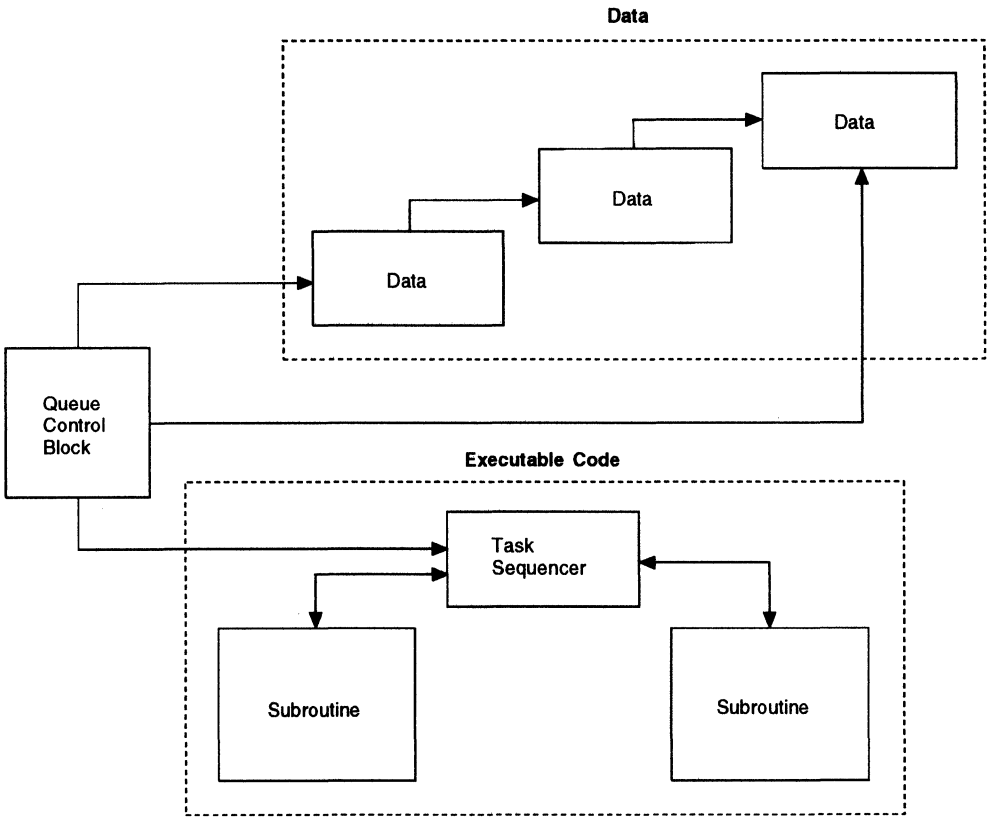


Figure 2. NCP Task Schematic

Figure 3 on page 8 shows an example of a task within NCP. In the example, the QCB is for requests that are not directed to a device. The queue is an input queue and the task is activated whenever any data is enqueued to the nondevice input queue.

You can schedule a level 5 task by establishing and enqueuing a QCB.

Task States

A task can be in any one of four logical states: active, pending, disconnect, or ready.

Active State: A task is in the active state when it is initiated by the task dispatcher. The task remains in the active state until it returns control to the supervisor through a SYSXIT macro. Only one task can be active in NCP at any time. An unconditional trigger can cause a task in the disconnect state to become active. A conditional or unconditional trigger can cause a task in the ready state to become active. A task can be active and pending at the same time.

Pending State: A task is in the pending state when its queue has been triggered and it cannot be dispatched by the task dispatcher because another task is currently in the active state. An unconditional trigger to a task in the disconnect state can cause the task to enter the pending state. A conditional or unconditional trigger to a task in the ready state can cause the task to enter the pending state. A PRELEASE macro issued by an active task can cause the task to enter the pending state. A task can be active and pending at the same time.

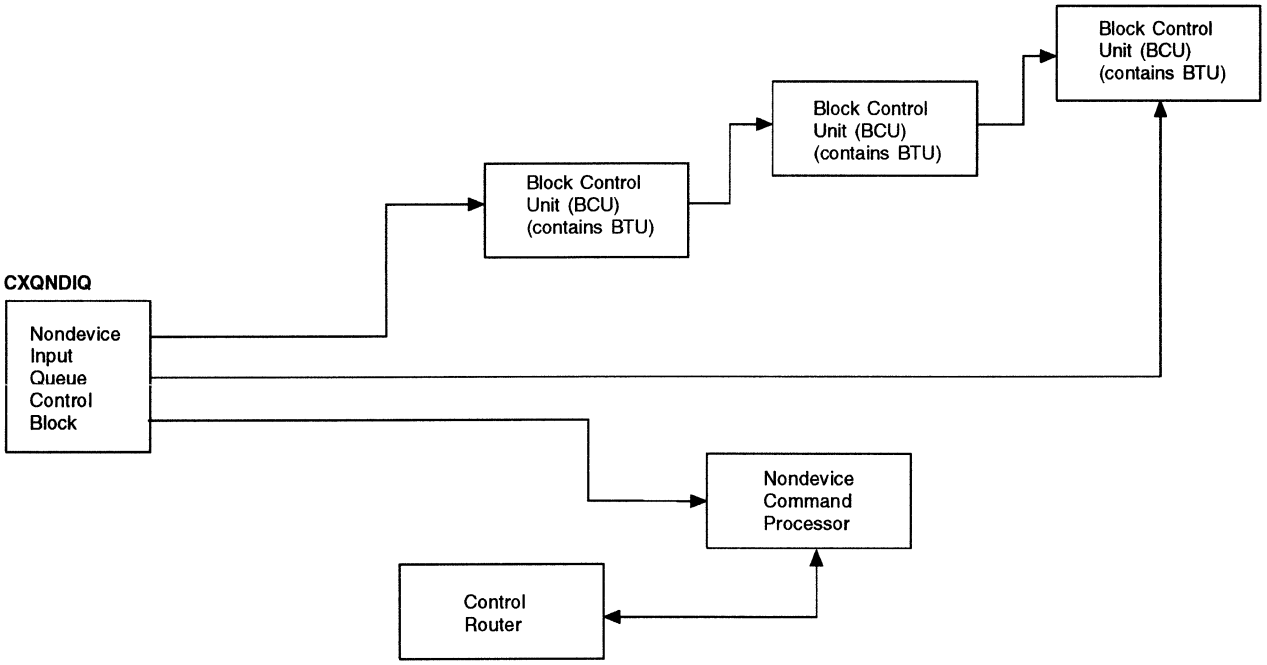


Figure 3. Example of an NCP Task for BSC and Start-Stop

Disconnect State: A task is in the disconnect state when it is not active, not pending, or not ready. Any number of tasks can be in the disconnect state at one time.

Ready State: A task is in the ready state when a QPOST macro has been issued for the queue governing task. A task can be ready and active or ready and pending. Any number of tasks can be in the ready state at one time.

Task-Scheduling Priorities

Tasks in NCP have one of four task-scheduling priorities: appendage, immediate, productive, and nonproductive. All tasks having the same priority are queued together.

Appendage: Appendage tasks have the highest priority in the system. They are dispatched on a first-in first-out (FIFO) basis from the appendage queue when the currently active task relinquishes control. Appendage tasks are generally initiated by the character service program at the end of a line I/O operation. However, they can also be initiated by the supervisor or by level 5 tasks.

Immediate: Immediate tasks have the second highest priority. Once processing for a line has started, all tasks necessary to initiate the I/O on the line are given the immediate priority.

Productive: Productive tasks have the third highest priority. A task is classified as productive if the end result of its execution is the initiation of output on either the channel or the communication line.

Nonproductive: Nonproductive tasks have the lowest priority in the system. A task is classified as nonproductive if it is not capable of starting I/O operations.

Tasks have scheduling priorities because appendage tasks are used to handle an

exceptional condition as soon as possible. Once a task associated with a line in the idle state receives control, the performance is better if all the tasks necessary to initiate the I/O on this line are dispatched in succession before dispatching tasks associated with any other lines. The immediate priority accommodates such tasks. Productive tasks have a high potential for freeing buffers and a low potential for allocating buffers. Nonproductive tasks have a low potential for freeing buffers and a high potential for allocating buffers. Hence, productive tasks should be executed before nonproductive tasks.

The CHAP macro can dynamically change the priority of a task.

NCP Supervisor

The routines that make up the NCP supervisor serve primarily as the interface between the routines in program level 5 and those in levels 2 and 3. It consists of the level 4 router control, the supervisor nucleus, the task dispatcher, and a group of service routines. The service routines are executed at the same level in which the macro was issued, except level 5.

The primary means of requesting the supervisor's services is by supervisor macros. When level 5 routines issue supervisor macros, they generate a supervisor call (SVC) code and cause a level 4 interrupt, called an SVC interrupt. The supervisory routines invoked to service SVC requests are executed at level 4.

When the supervisor macros are issued in levels 3 and 4, they generate a branch to the service routine that will handle the request. Level 2 supervisor macros expand and execute inline.

The supervisor can receive control through the following types of program controlled interrupts (PCI):

SVC	Requests supervisor service for level 5
Slowdown	Generates a system slowdown entry message
Dispatch	Schedules level 5 tasks
MOSS down	Processes MOSS down or offline request
CRP	Processes check-record-pool entry request.

See *NCP and EP Reference* for a detailed description of supervisor processing.

Level 4 Router Control

The level 4 router control determines the cause of the interrupt. If the interrupt is an SVC, it validates the SVC code and uses it as an index into the SVC vector table, located at the beginning of the supervisor nucleus. The table is used to pass control to the appropriate SVC macro decoder in the nucleus.

PCI requests are controlled by the PCIL4 macro and the level 4 router control block. The user issues a PCIL4 macro with the appropriate type of PCI coded as an operand.

Supervisor Nucleus

The supervisor nucleus is structured so that it can be entered through a level 5 SVC or through a level 4 or level 3 macro call. A level 5 SVC causes the interrupt handler to enter the nucleus through the SVC vector table, which in turn causes execution of the appropriate macro decoder. Each macro decoder has an entry point in the nucleus.

The macro decoder sets up the registers for the requested function and invokes the appropriate service routines to perform the function. When processing is complete, and the SVC is from the background, the nucleus exits; otherwise, it returns to the caller.

Supervisor macros issued in levels 4, 3, and in level 1 during initialization, generate a branch instruction. If the macro was coded with standard register usage, it branches directly to the appropriate service routine or routines. If the macro was coded with nonstandard register usage, the branch instruction is to the appropriate macro decoder in the supervisor nucleus, which in turn branches to the service routine or routines. For example, register 3 is standard for the *element address* operand of the DEQUE macro. If (3) is coded for this operand (indicating that the element address is in register 3), the branch instruction in the macro expansion is to the DEQUE service routine. If the element address is coded in any other register, the branch instruction generated by the macro is to the entry point in the supervisor nucleus. The nucleus then branches to CXADEQU.

When processing is complete, the supervisor returns to the instruction immediately following the macro.

After the appropriate service routine has been executed and the requested function completed, the macro decoder stores the output parameters (when nonstandard registers are used) or puts the output parameters into prespecified registers (when standard registers are used).

Supervisor Functions

The NCP supervisor's services are classified into four functional areas: task management, queue management, buffer management, and supervisory services. The following discussion describes these functions and the macros used to request them. The service routine that performs the requested function is shown in parentheses following the first mention of the macro. If no routine is indicated, the macro generates inline code, or it is processed in the supervisor nucleus.

Task Management

Task management includes dispatching tasks, manipulating task status, managing subroutine linkage, managing save area pools and chains, allocating save areas, and scheduling BSC and start-stop BHRs.

Dispatching Tasks: The task dispatcher determines the next level 5 task to be given control. If there is already an active task in level 5, the dispatcher schedules it for execution.

The task dispatcher receives control when all level 4 processing is complete. If the level 4 processing is the result of a PCI, control comes from the level 4 router control. If the processing in level 4 is the result of an SVC interrupt, control is from the supervisor nucleus.

When a task is dispatched, the level 5 interrupt address register (IAR) is loaded with the task's entry point, and register 2 in level 5 is loaded with the address of the active QCB. Also, the level 5 protection key is set to match the task storage key, and the task-active bit is set in the system mask.

Table 2 and Table 3 show the search order for the first nonempty dispatch queue.

Table 2. Search Order When Queue Not in Slowdown

	Appendage	Immediate	Productive	Nonproductive
Unconditional	1	4	7	10
Conditional	2	5	8	11
Normal	3	6	9	12

Table 3. Search Order When Queue in Slowdown

	Appendage	Immediate	Productive	Nonproductive
Unconditional	1	3	5	7
Conditional				
Normal	2	4	6	8

Manipulating Task Status: Task status manipulation is composed of seven services, each of which is specified by a macro.

- The TRIGGER macro schedules a task for execution by enqueuing the task input (or pseudo-input) QCB to the appropriate dispatching queue (appendage, immediate, productive, or nonproductive), thereby placing it in the pending state. TRIGGER macros are not accumulated in NCP; therefore, an attempt to trigger a task that has already been triggered is ignored. A conditional TRIGGER macro (TYPE=COND) is effective only if the task is in the ready state (that is, a QPOST macro has been issued for the task). If the task is not in the ready state, the TRIGGER macro has no effect. An unconditional TRIGGER macro is effective whether or not the task is ready.
- The CHAP macro changes the associated task entry point in a QCB or changes the scheduling priority, or both. If the task is in the pending state at the time its priority changes, it is removed from its current task dispatching queue and enqueued to the dispatching queue for the new priority. However, if the task is not in the pending state at the time its priority changes, then its scheduling flag is merely changed so that subsequent activation is appropriately scheduled.
- The SETEVNTL macro establishes the linkage between an event, represented by an event control block (ECB), and a task to be scheduled for execution upon completion of the specified event.

- The POST macro flags an ECB to indicate that an event is complete. If a SETEVNTL macro has been issued previously, the task specified in the macro is triggered, thereby rescheduling it for execution.
- The QPOST macro flags a task as being in the ready state and, therefore, available to be activated. Once a task is dispatched for execution it cannot be redispatched conditionally (through an ENQUE macro with ACTV=YES or through a TRIGGER macro with TYPE=COND) until a QPOST macro is issued for the task's QCB. If any data is enqueued to the task's input queue at the time the QPOST macro is issued, the task is automatically triggered and scheduled for execution.
- The SYSXIT macro indicates that the task is leaving the active state. In most cases, the dispatcher receives control to schedule the next pending task for execution. However, when a BSC and start-stop block handling routine (BHR) uses a SYSXIT macro, the BHR scheduler is invoked to schedule the next BHR in the block handler set for the appropriate point of execution.
- The ABEND macro posts an abnormal termination code in absolute storage location XDH at offset X'60' and issues an Output X'79' (request IPL) followed by an Output X'70' instruction (hard stop).

  This is done either when the supervisor or a level 5 task encounters a logical failure or when an unrecoverable hardware exception occurs.

Managing Subroutine Linkage: The subroutine linkage manager of task management provides the forward and backward linkage between a level 5 program and its subroutines. It also allocates and structures save areas for level 5 tasks and manages the level 5 save area pool.

Managing the Level 5 Save Area Pool: The level 5 save area pool consists of a chain of buffers leased by the supervisor to contain save areas for level 5 tasks. The supervisor leases the buffers one at a time as they are needed. When the first buffer is leased after IPL, a pointer to it is placed in extended halfword direct addressable HWE label SYSSPOOL. This buffer remains permanently allocated. Any subsequent buffers leased for the save area pool are chained to the first buffer but are then unchained and released when the save areas they contain are freed.

Each buffer leased for the save area pool is initialized by the supervisor to contain the count of available save areas in the buffer and a pointer to the next available save area. The count and pointer are updated each time a save area in the buffer is allocated or freed. If the count reaches 0, the supervisor leases a new buffer at the next request for a save area.

Managing Save Area Chains: A save area chain consists of all the save areas allocated for a single level 5 task. The supervisor allocates a save area and stores the registers of the calling routine in it each time a routine calls a lower-level routine. The supervisor frees the save areas in the reverse order as each routine returns control to its calling routine. Thus when the task has completed execution, all its save areas should have been freed.

Each save area contains a pointer to the previous save area and one to the next save area in the chain. The task's QCB contains a pointer to the last allocated save area. The supervisor provides these pointers when it allocates a save area.

The backward pointer of the first save area and the forward pointer of the last save area in a chain are always 0.

Allocating Dynamic Save Areas: When a level 5 routine needs a save area, it can request that the supervisor allocate it dynamically. The routine uses the CALL macro to do this, specifying ATTR=REENT and the registers to be saved as operands of the macro. The CALL service routine invokes the dynamic save area allocation routine. This routine allocates a save area from the save area pool. If there are no more available save areas, it issues a LEASE macro to allocate a new buffer. The CALL service routine then stores the user-specified registers in the save area, chains this save area to the previous save area, and passes control to the subroutine's entry point.

Each level 5 subroutine must begin with a SUBRTN macro. If the CALL macro specified ATTR=REENT, the SUBRTN macro generates an SVC. When the subroutine completes execution, it branches to its SUBRTN macro to return to the calling routine.

Allocating Static Save Areas: The supervisor provides two other ways for level 5 tasks to allocate save areas. If a routine uses either of these methods, the save area is generated inline in the code rather than being allocated from the save area pool. These save areas are static; that is, once allocated, they are never freed.

The calling routine can use the SAVEAREA macro to generate an inline save area. This macro generates a load-address instruction for each register to be saved and a branch instruction to the location just after the CALL macro that uses this save area. When the routine issues a CALL macro to call a subroutine, it specifies ATTR=NONREENT and provides the label of the SAVEAREA macro where the registers are to be saved. The CALL service routine then stores the registers in the load-address instructions generated by the SAVEAREA macro and passes control to the subroutine. When save areas are allocated in this way, the subroutine's SUBRTN macro generates a branch to the first load-address instruction in the SAVEAREA macro expansion. The load-address instructions are executed sequentially to restore the registers, and the branch instruction returns control to the next instruction to be executed in the calling routine.

The second method of allocating static save areas is available through the CALL macro alone. In this case, the user specifies ATTR=NONREENT and a list of the registers to be saved. The CALL macro generates a series of load-address instructions inline and stores the registers in the load-address instructions. The SUBRTN macro generates a branch back to the first load-address instruction in the CALL macro when the registers are to be restored.

The advantage of using the SAVEAREA macro comes when a routine contains several CALL macros. Under these circumstances, only one SAVEAREA macro is necessary for all the CALL macros. This requires less storage than if each CALL macro generated its own save area, because static save areas are permanently allocated when the code is assembled.

Scheduling BSC and Start-Stop Block Handling Routines: Block handling routine (BHR) scheduling consists of executing and terminating the BSC and start-stop block handling routines. The EXECBHR macro generates instructions to suspend normal task processing, lease a save area, save specified registers, and

pass control to the BHR controller. The controller determines the sequence of block handlers to be executed. As each BHR issues a SYSXIT macro, the BHR controller is called to determine the next block handler to be executed. When all block handlers have been executed, the registers are restored, and the allocated save area is freed. Control is returned to the task for resumption of normal task processing.

A BHR may end processing prematurely and present the BCU to the channel adapter I/O supervisor by using the ABORT macro.

The BHR dispatcher performs three functions in dispatching BHRs:

- When the routine initially receives control as a result of an EXECBHR macro, it sets up the registers of the issuing task and invokes the first BHR.
- When the routine receives control because of a SYSXIT macro, it invokes subsequent BHRs.
- The routine can also receive control as a result of an ABORT macro or as a result of normal block handler termination. In the latter case, it restores control to the task that first issued the EXECBHR macro.

This service routine receives control only from the supervisor nucleus because only level 5 code can issue EXECBHR macros. When the block handler ends, control is given to the dispatcher to determine the highest priority level 5 task to receive control.

Queue Management

All PIUs and BCUs in NCP are contained within a large array of common storage, but the elements in a particular queue are not necessarily contiguous within that storage. One element is chained to the next by a field in the ECB that contains a pointer to the next element in the queue. A QCB for each queue is maintained outside the array of buffer storage. Each element contains pointers to the first and the last elements in each queue.

A task may obtain a pointer to an element in a queue by causing the element to be removed from the queue by way of the DEQUE macro. A task may also obtain a pointer to an element without causing the element to be disconnected from the queue by way of the POINT macro.

First-in, first-out (FIFO) is the basic mode of queue manipulation in NCP. In the FIFO mode, elements are always connected to the end of a queue chain (ENQUE macro). Elements are always removed at the beginning of a queue chain (DEQUE macro).

An alternative mode of queue manipulation exists in NCP. In this mode a queue scan is initiated by a POINT macro and continued by an ADVAN macro. Elements are inserted at or removed from the desired position in the queue chain by INSERT and EXTRACT macros.

BCUs, PIUs, and QCBs are enqueued to system queues. A QCB cannot be chained into the same queue chain that it governs. The VALQCB macro is issued to validate that an element is a QCB. Queue management validates PIUs and BCUs to ensure that they are within the limits of buffer storage before adding them to a queue chain.

The MVQUE macro is used to move the entire contents of one queue to another queue.

Buffer Management

The free buffer pool is dynamic. You can remove buffers from the free buffer pool as they are needed to contain data using the LEASE macro. When the data in the buffers is no longer needed, use the RELEASE macro to reattach the buffers to the free buffer pool. If you specify that the data in the buffers is sensitive, zeros are inserted in the buffers when they are released.

You can chain buffers together by issuing the CHAIN macro or unchain buffers by issuing the UNCHAIN macro.

Buffer Lease Functions: Three types of buffer leases can be issued using the LEASE macro. They are CWALL, unconditional (UN), and conditional (COND). CWALL leases are satisfied, provided CWALL is not entered; unconditional leases are satisfied if there are enough buffers; and conditional leases are satisfied, provided NCP does not enter slowdown.

A CWALL type lease may be coded in two ways:

- Issuing a LEASE macro coded SUPV=CHARSERV results in an inline lease routine that leases one or more buffers at a time.
- Issuing a LEASE macro coded TYPE=CWALL results in a branch to the supervisor lease routine. This is valid only with SUPV=YES coding.

Unconditional leases are obtained by issuing a LEASE macro coded TYPE=UN and SUPV=YES or NO. This type of LEASE macro should only be used when the leasing routine could cause a deadlock if the lease is not satisfied. This is the only means of obtaining buffers below the CWALL threshold (with the exception of buffers that have been preleased).

Conditional leases are obtained by issuing a LEASE macro coded TYPE=COND and SUPV=YES or NO. This type of lease is used when satisfying the lease is not critical and the lease may be delayed until some buffers are released.

When buffers are released, they are put back onto the end of the free buffer chain and the current free buffer count is incremented by the number of buffers released.

Buffer Commit Functions: The purpose of a commit request operation is solely to control polling.

The COMMIT macro is issued to assure that buffers will be available for messages received on lines when the LEASEs are issued. For example, prior to sending a poll on a BSC line, NCP issues a COMMIT macro for the expected number of buffers needed to contain the message that may be received in response to the poll.

This macro supports conditional (COND), unconditional (CWALL), and precommit (PRECOMIT) COMMIT requests. BSC and start-stop line code and peripheral link code issue COND COMMIT requests that are satisfied if the system does not enter the pseudo-slowdown state. CWALL COMMIT requests are satisfied if the system does not enter the pseudo-CWALL state. You can issue the PRECOMIT COMMIT request to determine if a commitment in progress is being made.

If a commit request is satisfied, the line is polled in normal fashion (RR). If a commit request is not satisfied, the line is polled with an RNR, which only allows the terminal to respond with an acknowledgment. Thus, the commit operation prevents polling for data when NCP is in pseudo-slowdown or pseudo-CWALL.

The decommit routine makes committed buffers available.

Buffer Prelease Functions: The PRELEASE macro is used to assure that buffers will be available for level 5 tasks when the LEASE macro request is issued. The PRELEASE macro is issued at the beginning of the task by those tasks that expect to lease buffers. The number of buffers preleased is the number that are expected to be leased.

If the PRELEASE request is satisfied, this routine increments the system prelease count (SYSPRELC) by the number of buffers preleased, and decrements the free buffer count (SYSBPCBC) by the same amount. Although the buffers have not been leased (removed from the free buffer chain), they are totally inaccessible to any lease request, except for a LEASE request issued on behalf of level 5. If the PRELEASE request is not satisfied and is coded POST=YES, the task is queued onto one of the prelease dispatch queues.

This macro supports the conditional (COND) and the unconditional (UNCOND) PRELEASE macros. Tasks that are critical to keeping the data flow open (preventing a deadlock) issue PRELEASE macros coded TYPE=UNCOND. They are satisfied if the current free buffer count (SYSBPCBC) remains greater than or equal to 0. Unsatisfied UNCOND PRELEASE requests may be queued onto the unconditional prelease dispatch queue. Level 5 tasks that are not of such a critical nature issue PRELEASE macros coded TYPE=COND. They are satisfied if NCP is not in slowdown. Unsatisfied COND PRELEASE requests may be queued onto the conditional prelease dispatch queue.

Supervisory Services

Supervisory services include starting I/O on a communication line, starting output on the channel, scheduling a timer interrupt, obtaining the system time or time and data stamp, changing the timer QCB, and performing data manipulation, utility, and miscellaneous services.

Starting I/O Operations: The XIO macro causes a program-controlled interrupt (PCI) that initiates an I/O operation over the communication lines. One of the operands is a pointer to the line control block (LCB) or station control block (SCB). The XIO macro is also used to cause a program-requested interrupt (PRI) that initiates a channel output operation. One of the operands is a pointer to the PIU containing the data to be sent over the channel to the host.

Timer-Related Services: The SETIME macro is used by level 5 tasks to set a time interval (in an integral number of seconds) and to specify to the supervisor the address to be branched to when the interval has elapsed. The task that sets the time interval designates the task to receive control when the interval has elapsed.

The GETIME macro is used by level 5 tasks either to read the interval timer (which measures the number of seconds that have elapsed in the current 1.5 hour time period) or to move all or portions of the system time and date stamp to a specified location.

Data Manipulation Services: Use the following macros to invoke the data manipulation services of the supervisor:

COPYBCU	Copies the BCU work area and the 14 bytes of BTU control information from one buffer into a second buffer.
GETBYTE	Fetches a data byte.
PUTBYTE	Stores a data byte.
COPYPIU	Copies a PIU into new buffers.

Utility Services: The following macros generate inline code; use them to invoke the utility services of the supervisor:

LDM	Loads a series of registers from sequential locations in storage.
STRM	Stores a series of registers into sequential locations in storage.
SWAP	Exchanges the contents of two registers without using any storage areas.

Use the following macros for conditional branching:

BNH	Branch compare not high
BNL	Branch compare not low
BH	Branch compare high
BCR	Branch register if C latch
BZR	Branch register if Z latch
BNC	Branch if not C latch
BNZ	Branch if not Z latch
BM	Branch if minus
BP	Branch if plus
BMZ	Branch if minus or 0
BPZ	Branch if plus or 0
BNE	Branch if not equal
BNDH	Halfword branch on index

Use the following macros to shift the contents of registers:

ASHIFT	Shifts an address constant left or right.
SLL	Shifts a 22-bit register left a specified number of bit positions.
SLLB	Shifts a register byte left a specified number of bit positions.
SLLH	Shifts a 16-bit register left a specified number of bit positions.
SRL	Shifts a 22-bit register right a specified number of bit positions.
SRLB	Shifts a register byte right a specified number of bit positions.
SRLH	Shifts a 16-bit register right a specified number of bit positions.

Use the following additional utility macros as needed:

INHIBIT	Disables any specified level (from 1 to 5).
RESET	Enables any specified level (from 1 to 5).
ECBINIT	Initializes the status bytes of a specified event control block (ECB).

Miscellaneous Services: Other supervisory services are performed by the following routines:

RSLVRID service routine and RSLVNAD service routine: These routines convert network addresses into either the address of the resource control block or the address of the resource vector table (RVT) entry corresponding to that network address.

Resolve SNP service routine and resolve SSCP service routine: These routines return the pointer to the SSCP-NCP session control block (SNP) or the network address of the SSCP corresponding to the input parameter (SNP mask).

Pre-SNA system router: This routine establishes command processing priorities and enqueues the FID0 PIU received from the host channel interface to the appropriate processing task's input queue. This routine is called by the element routing function.

TPPOST service routine for BSC and start-stop: This routine, invoked by the TPPOST macro, disposes of a BTU after processing for it has been completed or has been discontinued due to an error. It invokes point 3 block handling routines, if any, and issues an XIO macro to the channel adapter I/O supervisor to return the BTU to the host.

RNSVC and SVLINK macros: The RNSVC and SVLINK macros generate the necessary code for SVC linkage. RNSVC generates the EXIT and DC instructions necessary for linkage from level 5 to the supervisor service routines (SUPV=NO). SVLINK generates the linkage to the supervisor service routines from code in level 3 or level 4 (SUPV=YES).

Most macros use RNSVC or SVLINK to provide the appropriate linkage code to the SVC decode modules. This routine supports conditional and CWALL commit requests. Conditional requests are accepted if the system is not in slowdown. If the system is in slowdown or if servicing the request puts the system in slowdown, the unsatisfied portion of the request is chained to wait for available buffers. CWALL requests are accepted if the system is not in buffer depletion mode. Unsatisfied CWALL requests are placed on a separate chain until buffers become available.

Interval Timer Routines

The interval timer routines are executed every 100 milliseconds at the occurrence of a program level 3 interval timer interrupt. The level 3 router gives control to the first timer routine and control is passed sequentially from one routine to another thereafter. The last routine returns control to the level 3 router.

Each of the interval timer routines performs one or more timer-oriented services, usually requested by some other portion of the network control program. See *NCP and EP Reference* for a description of system timer routines.

The communication line timer service routine scans each high-timer-resolution (BSC or SDLC) communication line and one-fifth of the low-timer-resolution (start-stop) lines during each 100 milliseconds. The communication line timer service routine determines from the line's CCB whether a new time-out has been set or an old time-out is still in effect.

For time-outs for user line control lines, the appropriate user routine is given control.

Using Customization Macros

IBM supplies NCP customization macros you can use to perform many of the functions required by line-control and programmed-resource routines. *NCP and SSP Customization Reference* describes these macros. This section explains how to use these macros in your routines.

Linkage Conventions

NCP uses register save areas to provide linkage between routines. Most macros use register 6 to point to the save area used by that macro. In general, you should not modify the contents of register 6. The SAVE, RESTORE, ROUTINE, and RETURN macros handle save areas. SAVE and RESTORE provide direct access to specific save areas. ROUTINE and RETURN provide a reentrant linkage structure for routines. See "Structured Process Blocks" on page 23 for more information on using ROUTINE and RESTORE, and *NCP and SSP Customization Reference* for more information on entrances and exits for user-written line control.

Register Usage

Except for register 6, you can use the registers in any manner you require. However, some macros work more efficiently with certain registers specified for certain keywords. Macro descriptions identify these registers.

Many macros also let you specify a set of work registers that do not need to be preserved across macro calls. Specifying work registers will usually improve performance of the macro-generated code.

Using Structured Programming Techniques

Structured programming makes your code easier to write and maintain, and it is strongly recommended that you make your custom code as structured as possible. NCP provides facilities to help you structure your code. These facilities include DO and IF macros and reentrant linkage structures for your subroutines.

IF, CASE, and DO Macros

The IF and CASE macros let you set up complex decision structures. The DO macros let you set up complex loop structures. Use these macro groups to give clear structure to your routines.

IF Macros

The IF macros are IF, THEN, ELSE, and ENDIF. ANDIF and ORIF macros are also provided, but use the AND and OR keywords of the IF macro to form a clearer structure.

The IF macro can perform four types of evaluation:

- Test CL, ZL
- Branch-on-bit
- Test-under-mask
- Comparison.

The test CL, ZL evaluation checks the C-latch or Z-latch. You must specify the true condition when you code the IF macro: on, off, zero, or not zero.

With branch-on-bit evaluation, you must specify a bit position of a register to be tested. As with the latch test, the true conditions are on, off, zero, or not zero.

The test-under-mask evaluation ANDs a specified register with the bit mask specified on the IF macro. If the true condition is specified as zero, the evaluation is true if all the bits in the result of the AND are zero. If the true condition is not zero, the result is true if any of the result bits are not zero.

The comparison evaluation compares two registers using the specified comparative operator. Operators are greater than, less than, equal to, not equal to, not greater than, not less than, greater than or equal, and less than or equal.

IF macros can be combined to form more complex evaluations such as `IF (A=2) AND (B<4)`. The best way to do this is by using the AND and OR keywords to link IF macros together. For example, to create the evaluation `IF ((A OR B) AND (C OR D))`, you would code the IF macros shown in the following example.

```
IF A,(OR)
IF B,AND
IF C,(OR)
IF D
```

The IF macros are evaluated in two steps. First, the keywords are evaluated from right to left; parentheses are placed around successive pairs of keywords connected by operators in parentheses. Second, the keywords are evaluated from left to right and parentheses are placed around remaining pairs of keywords. Thus, on the first pass, the preceding macros evaluate to `IF (A OR B) AND (C OR D)` and on the second pass become `IF ((A OR B) AND (C OR D))`.

Figure 4 on page 21 shows how the IF macros can form a decision structure.

```
IF RA,EQ,RB,H,AND   If register A is equal to register B
IF RC,GT,RD,H       And if register C is greater than register D
    THEN            Then do.
      -       -
      -       -
      -       -
    ELSE            Else do.
      -       -
      -       -
      IF RC,NE,RD     If register C is not equal to register D,
         THEN           Then do.
           -       -
           -       -
         ELSE           Else do.
           -       -
           -       -
      ENDIF           Must end code under ELSE
ENDIF               Ends whole IF structure
```

Figure 4. Sample IF Macro Structure

CASE Macros

The CASE macros (CASE, CASEIF, CASENTRY, CASEXIT, and ENDCASE) provide multibranch control structures that let you select from many possible alternatives without using complex IF structures. As Figure 5 on page 22 shows, the CASE macro uses the same evaluation types that IF uses.

```
CASENTRY
    -
    -
    CASEIF RA,EQ,RB,CASE=XYZ   If reg A is equal to reg B, do case XYZ.
    CASEIF RA,EQ,RC,CASE=WXY   If reg A is equal to reg C, do case WXY.
    CASEIF RA,EQ,RD,CASE=WWX   If reg A is equal to reg D, do case WWX.
    CASE=ZZZ                   Otherwise, do case ZZZ

CASEXIT                        All cases return here.


⋮

XYZ CASE
    -
    -    Case XYZ
    -
    -
    ENDCASE

WXY CASE
    -
    -    Case WXY
    -
    ENDCASE

WWX CASE
    -
    -    Case WWX
    -
    ENDCASE

ZZZ CASE
    -
    -    Case ZZZ
    -
    ENDCASE
```

Figure 5. Sample CASE Macro Structure

All cases, including the fall-through case, return to the CASEXIT point when they are done executing.

DO Macros

The DO macros (DOWHILE, DOUNTIL, LEAVEDO, and ENDO) let you create loop structures. DOWHILE loops check the looping condition before the loop is entered; DOUNTIL loops check the condition at the end of the loop. DOUNTIL loops are always executed at least once, but DOWHILE loops may not be executed at all if the condition is not true on the first invocation.

Use the same conditions for the DO macros as for the IF macros, including the same use of optional AND and OR keywords. The DO macros evaluate the following conditions:

- No operation
- DO X by Y
- Branch-on-count
- Test CL, ZL
- Branch-on-bit

- Test-under-mask
- Comparison.

The no-operation condition does no evaluation at all. It lets you put the loop test inside the loop. You must use the LEAVEDO macro to exit the loop.

The DO-X-by-Y condition performs the loop while *X* is greater than zero. The *X* value is decremented by *Y* each time. If you use it, this condition must be on the first DO coded in a group of DO macros and can be used only once per DO group.

The branch-on-count condition decrements the contents of a register until it is equal to zero. A DO using this condition cannot be linked to other DOs, although it can be nested inside a DO loop or have other DOs nested inside it. The test is assembled at the end of the loop for maximum efficiency. The branch distance is limited because it uses the BCT instruction.

Figure 6 shows how a DOWHILE loop can be used.

```
DOWHILE RA,EQ,RB,H,AND   While register A is equal to register B
DOWHILE RC,GT,RD,H       And register C is greater than register
                         D, execute loop.
    -       -
    -       -
    -       -
ENDDO                    Ends DO loop.
```

Figure 6. Sample DO Macro Structure

Structured Process Blocks

Structured process blocks provide reentrant subroutine linkage for your routines. A process block consists of a ROUTINE-RETURN-PERFORM macro sequence. The ROUTINE macro sets up the entry point of the routine to be called, saves all the designated registers in the current save area, and updates the save-area pointer (register 6) to point to the next save area. The RETURN macro updates the save-area pointer to point to the previous save area, restores the registers saved in the save area, and returns to the calling routine. The PERFORM macro establishes linkage to the instruction following the PERFORM macro and calls the routine.

Code ROUTINE at the beginning of your routine and RETURN at the end. You can then use PERFORM to execute that routine from anywhere in another routine.

If you need less structured linkage, you can use the SAVE and RESTORE macros to preserve registers across routine calls.

Segmentation

Structured programming separates a large program into smaller segments consisting of one main, high-level segment and other low-level segments. For the strictest structuring, each segment or subroutine should do only one well-defined task. As a rule of thumb, a segment should occupy less than two printer pages in the unexpanded (PRINT=NOGEN) listing. Lower-level segments may call other

lower-level segments; however, they always return to the calling segment. Each segment has only one entry point and one exit point.

Module Listings

Maintain two listings for structured modules: unexpanded (PRINT=NOGEN) and expanded (PRINT=GEN). The unexpanded form, where no macros are expanded, is much easier to read and understand because the expansions of the macros obscure the indentation established for the macro calls and the instructions outside the macros. The expanded versions of the listings are useful for direct comparison with storage dumps and for detailed program examination.

Using Binary Trees

You can use a set of general-purpose macros to manipulate control blocks organized into binary trees. This book assumes you are familiar with binary tree principles.

The BT macros (BTECHECK, BTDELETE, BTINSERT, and BTSEARCH) permit you do any of the following:

- Check whether or not a tree has any nodes
- Add nodes to a tree
- Delete nodes from a tree
- Search a tree for a specific node.

Two control blocks, the search-tree header block (SHB), which forms the anchor of a tree, and the search-tree element block (SEB), which forms an element or node of the tree, build NCP's binary trees. These control blocks can be embedded in other control blocks.

The SHB contains a pointer to the root node (the SEB) of a tree. If the SHB is embedded in another control block, the first byte of the SHB contains the offset to the SHB from the start of the embedding block.

The SEB contains pointers to its left and right subtrees plus a search-key field that uniquely identifies the node. In standard NCP usage, the left and right subtree pointers act as thread pointers if no subtrees are attached to a node. The thread pointers point up to the SEB parent node. The first byte of the left subtree pointer (SEBLLINK) contains the SEB flags. If the SEB is embedded, the first byte of the right pointer (SEBRLINK) contains the offset to the SEB from the start of the embedding control block.

NCP expects balanced binary trees, and the insert and delete macros automatically balance trees when nodes are added or deleted. The SEB flags indicate whether or not a subtree is balanced. See *NCP and EP Reference Summary and Data Areas*, Volume 1, for the exact meanings of the SEB flags.

Figure 7 shows a simple balanced binary tree using an SHB and an SEB. The embedded control blocks are not shown.

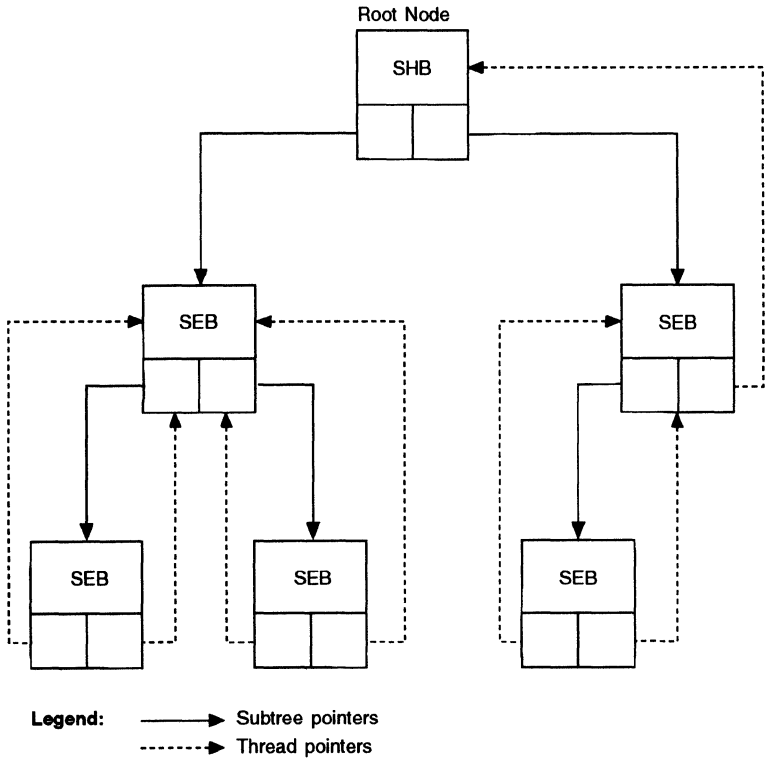


Figure 7. Simple Binary Tree

Using Gateway Exits

The gateway-user accounting exit, GWAEXIT, allows you to code an accounting exit routine to collect information about cross-network session traffic. This exit receives control whenever a PIU crosses a network boundary. GWAEXIT can receive control for all cross-network sessions, which include LU-LU sessions, SSCP-SSCP sessions, and FID0 sessions.

NCP supports gateway-session accounting using NetView* Performance Monitor (NPM). This may eliminate the need to write your own gateway-session accounting routines. See *NCP, SSP, and EP Resource Definition Guide* and *NCP, SSP, and EP Resource Definition Reference* for information on including this function.

NCP provides NPM accounting management for IBM special products or user-written code. See "Using NPM Session Accounting" on page 67 for information about session accounting for IBM special products or user-written code and "Using NPM Data Collection with Programmed Resources" on page 99 for information about data collection.

Coding the User Accounting Exit

You must use the UPARMS and URETURN macros in your accounting exit routine. UPARMS helps gather information about cross-network path information units (PIUs) and URETURN provides a return linkage to NCP.

Use the UPARMS macro to obtain the following information:

- Pointers to the PIU that is crossing the network boundary
- The virtual route on which the PIU is traveling
- The resource name of both logical units
- The networks in which the logical units reside
- The addresses of the logical units
- A time and date stamp.

The URETURN macro must be the last macro in the accounting exit. URETURN passes control back to NCP. The sample CSECT in the routine shown in "Sample Gateway Exit Routine" on page 27 shows how to use UPARMS and URETURN.

Requirements for a User Accounting Exit

GWAEXIT must be specified on the BUILD definition statement in the generation definition with a user routine present in the USERLIB. Specify the name of the entry point of your accounting exit routine on the GWAEXIT keyword of the BUILD definition statement. This name is the name of the entry point in the module identified by the INCHI and ORDHI keywords of the GENEND definition statement.

It is your responsibility to move the accounting information from NCP to the host for processing. You can do this by coding an NCP programmed resource to send the information to the host and by coding a VTAM* application program to receive and process the information. See *VTAM Programming* for information on VTAM application programming.

You can declare storage areas in the gateway NCP to use for collecting session information. There is no limit (logically) to how much storage you can use, but you must leave enough storage for buffers and cross-network resource control blocks.

Sample Gateway Exit Routine

The following routine demonstrates how to code a gateway exit.

```
  SAMPLE CSECT

         EXTRN  CXTVVT    FOR RSLVVVTI MACRO
         EXTRN  CXDNETID  FOR RSLVNET MACRO
         LA     R1,SAVPTR  GET ADDRESS OF USER'S CONTIGUOUS STORAGE
         USING PARMS,R1   ESTABLISH ADDRESSABILITY TO USER PARM AREA
*****************************************************************
* GET PRIMARY OPERAND FIELDS AND STORE INTO CONTIGUOUS STORAGE *
*****************************************************************
         UPARMS PLUNAME=(2),PNETID=(3),PLUSIDE=(4,5),PSVVTI=(6)
         ST   R2,PNAME           STORE PLU NAME POINTER
         ST   R3,PNETID          STORE PLU NETID POINTER
         STH  R4,PREALEAD        STORE PLU REAL ELEMENT ADDRESS
         STH  R5,SALIASEA        STORE SLU ALIAS ELEMENT ADDRESS
         STH  R6,PSVVTI          STORE PLU SIDE VVTI
*****************************************************************
* ISSUE RSLVVVTI USING VVTI FROM UPARMS AS INPUT TO GET THE    *
* SUBAREA PART OF THE PLU REAL ADDRESS                          *
*****************************************************************
         RSLVVVTI VVTI=(6),HISUBA=(2),LOSUBA=(4),WREGA=(5)
         STH  R2,PREALSAH  STORE PLU REAL SUBAREA HIGH HALFWORD
         STH  R4,PREALSAL  STORE PLU REAL SUBAREA LOW HALFWORD
*****************************************************************
* ISSUE RSLVNET USING NETID FROM UPARMS AS INPUT TO GET THE    *
* SUBAREA PART OF THE SLU ALIAS ADDRESS                         *
*****************************************************************
         RSLVNET INPUT=(NETID,3),NCP=NO,HISUBA=(2),LOSUBA=(4),WORKR=(5)
         STH  R2,SALIASSH  STORE SLU ALIAS SUBAREA HIGH HALFWORD
         STH  R4,SALIASSL  STORE SLU ALIAS SUBAREA LOW HALFWORD
*****************************************************************
* GET SECONDARY OPERAND FIELDS AND STORE INTO CONTIGUOUS STORAGE *
*****************************************************************
         UPARMS SLUNAME=(2),SNETID=(3),SLUSIDE=(4,5),SSVVTI=(6)
         ST   R2,SNAME           STORE SLU NAME POINTER
         ST   R3,SNETID          STORE SLU NETID POINTER
         STH  R4,SREALEAD        STORE SLU REAL ELEMENT ADDRESS
         STH  R5,PALIASEA        STORE PLU ALIAS ELEMENT ADDRESS
         STH  R6,SSVVTI          STORE SLU SIDE VVTI
*****************************************************************
* ISSUE RSLVVVTI USING VVTI FROM UPARMS AS INPUT TO GET THE      *
* SUBAREA PART OF THE SLU REAL ADDRESS                          *
*****************************************************************
         RSLVVVTI VVTI=(6),HISUBA=(2),LOSUBA=(4),WREGA=(5)
         STH  R2,SREALSAH  STORE SLU REAL SUBAREA HIGH HALFWORD
         STH  R4,SREALSAL  STORE SLU REAL SUBAREA LOW HALFWORD
```

Figure 8 (Part 1 of 2). Sample Gateway Exit

```
*********************************************************************
* ISSUE RSLVNET USING NETID FROM UPARMS AS INPUT TO GET THE      *
* SUBAREA PART OF THE PLU ALIAS ADDRESS                          *
*********************************************************************
         RSLVNET INPUT=(NETID,3),NCP=NO,HISUBA=(2),LOSUBA=(4),WORKR=(5)
         STH  R2,PALIASSH  STORE PLU ALIAS SUBAREA HIGH HALFWORD
         STH  R4,PALIASSL  STORE PLU ALIAS SUBAREA LOW HALFWORD
*************************************************************************
* GET TOD, PIU, AND VRTPF OPERAND FIELDS AND STORE IN CONTIGUOUS STORAGE*
*************************************************************************
         LA   R4,TOD-PARMS
         AR   R4,R1
         UPARMS VRTPF=(3(1)),PIU=(2),TOD=(4)
         STC  R3LO,VRTPF          STORE VRTPF NUMBER
         ST   R2,PIUP             STORE PIU POINTER
         URETURN
SAVPTR   DC   14A(0)
         DC   CL1'0'
*************************************************************************
PARMS    DSECT
PNAME    DS   A       POINTER TO PLU NAME
PNETID   DS   A       POINTER TO PLU NETID
PREALSAH DS   H       PLU REAL ADDRESS - HIGH SUBAREA HALFWORD
PREALSAL DS   H       PLU REAL ADDRESS - LOW SUBAREA HALFWORD
PREALEA  DS   H       PLU REAL ADDRESS - ELEMENT HALFWORD
PALIASSH DS   H       PLU ALIAS ADDRESS - HIGH SUBAREA HALFWORD
PALIASSL DS   H       PLU ALIAS ADDRESS - LOW SUBAREA HALFWORD
PALIASEA DS   H       PLU ALIAS ADDRESS - ELEMENT HALFWORD
PSVVTI   DS   H       PLU SIDE VVTI
*************************************************************************
SNAME    DS   A       POINTER TO SLU NAME
SNETID   DS   A       POINTER TO SLU NETID
SREALSAH DS   H       SLU REAL ADDRESS - HIGH SUBAREA HALFWORD
SREALSAL DS   H       SLU REAL ADDRESS - LOW SUBAREA HALFWORD
SREALEA  DS   H       SLU REAL ADDRESS - ELEMENT HALFWORD
SALIASSH DS   H       SLU ALIAS ADDRESS - HIGH SUBAREA HALFWORD
SALIASSL DS   H       SLU ALIAS ADDRESS - LOW SUBAREA HALFWORD
SALIASEA DS   H       SLU ALIAS ADDRESS - ELEMENT HALFWORD
SSVVTI   DS   H       SLU SIDE VVTI
*************************************************************************
TOD      DS   2F      TIME OF DAY
PIUP     DS   A       PIU POINTER
VRTPF    DS   CL1     VIRTUAL ROUTE TRANSMISSION PRIORITY FIELD
*************************************************************************
             XCXTEQU
             XCXTUIC
             XCXTNLX
             XCXTNIX
             XCXTNLB
             XCXTNIB
             XCXTPIU   TYPE=4,SCRUEQU=(ACTCDRM,REQACT,NOTIFY)
             XCXTRCB
             XSYSEQU
             XSYSABNS
             XCXTNVT
             XCXTVVT
```

Figure 8 (Part 2 of 2). Sample Gateway Exit

Using Block Handlers

Block handling is the optional message processing of start-stop and binary synchronous communication (BSC) message data within the communication controller. NCP can process message data from the host processor before sending it to a start-stop or BSC station or it can process message data received from a station before sending it to the access method. Processing is possible only when the message data between controller and station is transmitted in network control mode.

You can add your own block handling routines to NCP in addition to the IBM-supplied block handlers that provide automatic text correction and that can insert date and time information into messages. User-written block handling routines permit you to add certain data-handling functions to NCP. These routines typically examine and manipulate incoming or outgoing data contained in NCP buffers.

If you understand NCP and your access method, adding such routines to NCP has little likelihood of disrupting the NCP code. However, adding routines that perform more complex functions, such as leasing and releasing buffers or scheduling I/O operations, requires a thorough understanding of the internal operation of NCP and the access method. Approach the adding of such routines with care to avoid disrupting NCP logic. See *NCP, SSP, and EP Resource Definition Guide* and *NCP, SSP, and EP Resource Definition Reference* for information on how to define block handling routines in the generation definition. See *NCP, SSP, and EP Generation and Loading Guide* for information on including your block handlers during NCP generation.

Your block handler processes data at one of three points.

- Point 1 is between the access method and NCP. At this point the block handler gets data before NCP has determined that a connection to the destination device is available.
- Point 2 is between NCP and the link station. At this point the block handler gets outgoing data after NCP has determined that a link is available but before the data is put on the link. Incoming data is processed as it comes from the link and while the link is being held for the device.
- At point 3, the block handler gets incoming data after the link for the device has been released.

Inserting Date and Time

NCP can insert the current date, time of day, or both into message blocks it receives from the access method over the network control subchannel or from a station over a line in network control mode.

The date can be in any of four formats:

- Month/day/year (10/21/94)
- Year/month/day (94/10/21)
- Day/month/year (21/10/94)
- Year followed by day of year (94.295).

The format for the time of day is hh.mm.ss (hours, minutes, seconds). The continental (24-hour) form is used. For example, 09.17.25 and 21.17.25 represent 9:17:25 a.m. and 9:17:25 p.m., respectively. An EBCDIC blank character precedes each format.

The date and time can be placed in the first block of each message or transmission, or in every block of the message or transmission.

Use the DATETIME definition statement to define this type of block handler.

Correcting Text Automatically

Automatic text correction is an editing function that lets NCP replace text entered incorrectly from a terminal keyboard with the corrected characters that the terminal operator subsequently sends. NCP does this by scanning each block for predefined characters called text-canceling characters. NCP removes each text-canceling character it finds and one preceding text character from the block. For example, if the program finds a sequence of 3 canceling characters, it removes these 3 canceling characters plus the 3 immediately preceding text characters.

A terminal operator might, for instance, enter COMMUNCIATE and, seeing that it is misspelled, enter 5 backspace characters to back up to the first wrong character. The operator then reenters the corrected characters. For example:

```
COMMUNCIATE bksp bksp bksp bksp bksp ICATE
```

If you have specified the text-correction option and designated *backspace* as the text-canceling character, the text-correction block handling routine removes the 5 backspace characters and CIATE. The remaining characters form the correctly spelled word COMMUNICATE.

The text-canceling character does not have to be a backspace character. Any other character (except a line control character) may be used if it is not used in any other way within message text. For example, if the slash (/) is the character chosen and a keyboard operator enters ATLANITC///TIC, the text correction block handling routine corrects the word to ATLANTIC.

Use the EDIT definition statement to specify the text-canceling character.

Writing Block Handling Routines

Block handling routines let you add certain data-handling functions to NCP for start-stop and BSC lines to NCP. These routines typically examine and manipulate incoming or outgoing data contained in NCP buffers. If you have a good general understanding of NCP and the access method, you can add such routines to the program with little possibility of disrupting NCP code. However, adding routines that perform more complex functions, such as leasing and releasing buffers or scheduling I/O operations, requires a thorough understanding of the internal operation of NCP and the access method. Be careful to avoid disrupting NCP when adding this type of routine.

Code a user-written block handling routine using the communication controller assembler language and assemble it using the controller assembler. Specific rules and guidelines to observe in coding user-written block handling routines follow. In the list, the term *macro invocation* refers to 3705 Assembler Language. See *NCP and SSP Customization Reference* for descriptions of the macro formats and details for use of supervisory macros.

- All registers can be used in a user-written block handling routine. Before passing control to a user-written routine, NCP saves all registers. When receiving control from the routine, it restores all registers.
- At entry to a user-written block handling routine, register 2 points to the queue on which the block being handled is enqueued.
- A POINT (2), (3) macro invocation causes the address of the block at the head of the queue to be returned in register 3.
- A SCAN macro invocation scans the text in chained NCP buffers containing the block being processed.
- The DEQUE, ENQUE, and INSERT macro invocations dequeue, enqueue, and insert the block's address that was returned by the POINT macro invocation.
- The LEASE macro invocation obtains NCP buffers. Use a RELEASE macro invocation to release any buffers thus obtained.
- The SYSXIT macro invocation returns control from the user-written block handling routine to NCP.
- If a user-written block handling routine is run for more than one BSC device or telecommunication link, code it so that it is serially reusable.
- If the user-written routine changes the amount of message text accompanying a basic transmission unit (BTU), the routine must update the BCUTLEN field of the BTU and the data count fields of the buffer prefix areas.
- Be sure user-written block handling routines do not modify any part of the first 34 bytes (the BTU) of a header buffer or the first 4 bytes of any other buffer.
- Logic errors encountered in user-written block handling routines can cause NCP to abend.

Associating Block Handling Routines with Stations

The application requirements determine how NCP processes messages before sending them to the network or the access method. The requirements may vary for different stations or for different parts of a station. For example, you may want to provide the text-correction function for messages entered from a terminal keyboard but not for messages received from a paper tape reader; or you may want to insert date and time information in messages received from station A but not in those received from station B.

NCP definition statements provide a way to group individual block handling routines into block handlers and to combine block handlers into block handler sets. Block handler sets can then be associated with individual stations or station parts.

Each block handler within a set can be executed at a different logical point in the flow of message data through NCP. For instance, one block handler in the set can be executed immediately when a message arrives from the host processor

(point 1). Another block handler in the same set can insert routines that process after the program has obtained use of a line (point 2). This block handler can also insert routines that process message data from a station before NCP releases the line over which it received the data. A third block handler in the set can be assigned to process message data received from a station after NCP has released the line (point 3).

Using IBM 3745 Hardware Features

Your customized code can use the IBM 3745 Communication Controller's ability to add and remove line adapters and to swap line and channel adapters between central control units (CCUs). To use these features, you must either have control blocks that are compatible with NCP's control blocks or write your own routines to handle these features.

Note: Various models of the IBM 3745 have different hardware features. The functions available depend on the hardware you are using as well as the software support you have. Refer to the appropriate hardware manual to determine which functions your IBM 3745 supports.

This section describes the keywords that enable you to use the IBM 3745 features and the functional changes you must make to your code for these features. See "User Adapter Control Block" on page 52 for the fields you must have in your control blocks to use NCP services for the IBM 3745 features.

Generation Keywords for Customization

Two keywords on the GROUP definition statement affect customized code. Those keywords are COMPOWN and COMPTAD. Coding YES on either COMPOWN or COMPTAD indicates that certain fields in the UACB for that group of lines meet compatibility requirements so that NCP can perform the IBM 3745 functions for those lines. "Control Blocks for User Line Control" on page 51 explains the fields affected. See *NCP, SSP, and EP Resource Definition Reference* for more information on coding these definition statements.

The UGLOBAL keyword on the GENEND definition statement lets you code the entry points of global routers that handle IBM 3745-specific functions.

COMPOWN

Coding COMPOWN=YES indicates that your UACB is compatible for switchback, LA disconnect, and fallback. The switchback and LA disconnect processors can check the control blocks to determine if the lines have SSCP owners and are active. The fallback processor checks the control blocks to deactivate a line that will be switched back to the other CCU.

When a switchback is requested from the maintenance and operator subsystem (MOSS) console, and you have coded YES on COMPOWN, NCP verifies that lines installed on adapters to be switched have been freed by the SSCP owners. If the switchback is forced by the MOSS operator, active lines installed on adapters to be switched are deactivated.

COMPTAD

Coding COMPTAD=YES indicates that your UACB is compatible for the following:

- NCP initialization
- Configuration data set (CDS) update
- Bus switching initialization of the TA and TD bus address
- Adapter information table (AIT) index initialization.

User-controlled lines that are defined without an address (ADDRESS=NONE on the LINE definition statement) must also specify COMPTAD=NO if the lines are to be activated. This line is not a real line and will not be associated with a line adapter. Therefore, it will not have a bus address (TA/TD).

Required Functional Changes for IBM 3745

This section details the changes you must make to your code to use the IBM 3745 features.

Changes to Level 1

The following changes to NCP level 1 may require changes to your code:

- Box event record (BER) formats
- AST and MCT references
- Level 1 subroutine CXBFNDBD.

BER Formats: Because the level 1 BER formats are new, you may need to change any BERs that are built to correspond to the new formats that NCP builds.

AST and MCT References: If any of your code refers to the AST or MCT control block, it must be changed to access the adapter information table (AIT) for the same information, because the AST and MCT have been removed for the IBM 3745.

CXBFNDBD: If your code currently uses NCP level 1 subroutine CXBFNDBD, you must change your code so that it does not rely on this subroutine, because CXBFNDBD does not exist in IBM 3745 level 1 support code.

User Parameter and Status Area for IBM 3745 Functions

The NEOG user parameter and status area is an interface between NCP and your router modules for IBM-specific 3745 functions. You code the entry points of your routers on the UGLOBAL keyword of the GENEND definition statement. These routers get control each time an IBM 3745 operation, such as add line adapter, is performed for a user-controlled resource. If more than one entry point is coded on the UGLOBAL keyword, the entry points are given control serially for a specific operation. Your routers get control from NCP for these operations with the following parameters:

Register 1	Pointer to NEOG parameter and status area
Register 6	Save-area pointer
Register 7	Return address.

Your routers should then check the operation field (NEOGOP) of the NEOG parameter and status area, and either process that operation or route control to a user-written module specific to that operation. Parameters can also be supplied from

NCP in the NEOG area; currently only NEOGP1 and NEOG4P1 are used as additional parameters.

Even though there is more than one user parameter and status area, it makes no difference to your routers because register 1 always supplies a pointer to the appropriate parameter and status area.

Figure 9 shows the layout of the NEOG user parameter and status area.

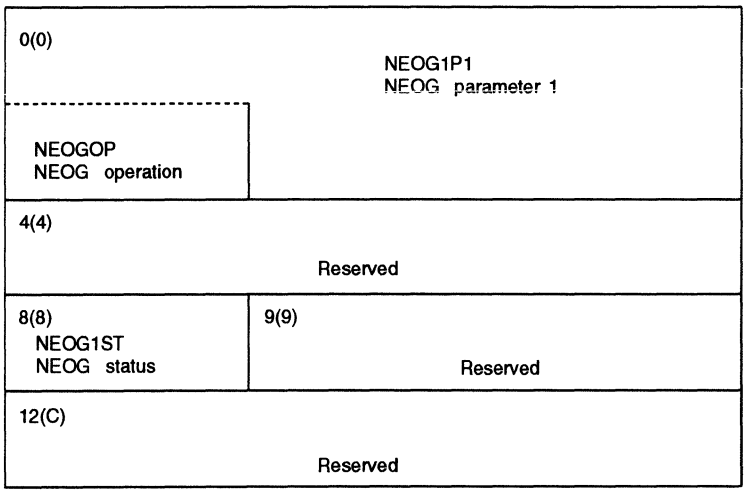


Figure 9. NEOG User Parameter and Status Area

This user parameter and status area is used only for the functions specified in Table 4. The use of NEOG does not affect any previously used methods of giving control to user-written code. Table 4 shows the functions that use the user parameter and status area and their operation codes.

Table 4 (Page 1 of 2). NEOG Operations and Parameters

Operation	NEOGOP Value	NEOGP1 Data	NCP Level
User fallback	X'01'	AIT address	4
User switchback	X'02'	AIT address	4
Delete port	X'03'	AIT entry address	4
Add port	X'04'	AIT entry address	4
Disconnect line adapter	X'05'	AIT entry address	4
Connect line adapter	X'06'	AIT entry address	4
Add channel adapter	X'07'	AIT entry address	4
Delete channel adapter	X'08'	AIT entry address	4
Disconnect channel adapter	X'09'	AIT entry address	4
Connect channel adapter	X'0A'	AIT entry address	4
CACM level 1 error	X'19'	AIT entry address	1
CACM disconnect channel-adapter-in-use check	X'10'	AIT entry address	4

Table 4 (Page 2 of 2). NEOG Operations and Parameters

Operation	NEOGOP Value	NEOGP1 Data	NCP Level
CACM initiate disconnect channel adapter	X'11'	AIT entry address	4
CACM disconnect channel adapter complete	X'12'	AIT entry address	4
CACM connect channel adapter initialize	X'13'	AIT entry address	4
CACM connect channel cancel	X'14'	AIT entry address	4
CACM update CDS insert channel adapter	X'15'	AIT entry address	4
CACM update CDS delete channel adapter	X'16'	AIT entry address	4
CACM update CDS change channel adapter	X'17'	AIT entry address	4
CACM autoselection chain insert	X'18'	AIT entry address	4

AIT address refers to the beginning of the AIT control block. AIT entry address refers to a particular adapter's AIT entry in the AIT control block.

NCP Initialization and Update Configuration Data Set

Because of the flexible relationship between ports and adapters, NDF does not generate the TA and TD address for a line. NCP initialization determines the appropriate TA and TD values for each line from the CDS passed from MOSS; it then initializes the appropriate line control blocks for NCP, the partitioned emulation program (PEP), and the user-controlled, compatible ports. The TA and TD values are also set as part of the UPDATE CDS function. NCP initialization and update CDS code ignores incompatible user-controlled ports.

If your control blocks are compatible and you code an entry point on the INIT keyword of the GENEND definition statement, you can then set the TA and TD values for your lines at initialization. These values may be set as part of an update CDS operation if you coded an entry point on the UGLOBAL keyword of the GENEND definition statement.

The user parameter and status area acts as the interface when control is passed to your code for an update CDS operation. This parameter and status area indicates the reason for passing control (delete line adapter, delete ports, add line adapter, or add ports) and provides pertinent information for the function. The code at the entry points specified on the UGLOBAL keyword of the GENEND definition statement must route control to the appropriate user routine based on information passed in the user parameter and status area.

Disconnect Line Adapter

When the disconnect-line-adapter function is invoked, your code may then relinquish control of the lines on the adapter being disconnected. This occurs if you specify an entry point on the UGLOBAL keyword of the GENEND definition statement. The disconnect-line-adapter function can be either conditional or unconditional.

Disconnect line adapter is a two-stage process. First, lines are checked for SSCP owners. If there are none, NCP proceeds with the disconnection procedure and tells MOSS to reset the LA. Second, if the MOSS console operator forces the disconnect-line-adapter function (even if some lines were found to have SSCP owners) or if an unconditional disconnect-line-adapter function is requested, NCP proceeds with the disconnection procedure, bypassing the first stage of the disconnect-line-adapter function (the check for SSCP owners).

When a conditional disconnect line adapter is requested and your control blocks are specified as compatible, NCP verifies that the user-controlled lines are not owned (the first stage of disconnect line adapter) before allowing the disconnection to occur. However, if a conditional disconnect line adapter is requested and your control blocks are not specified as compatible, NCP does no checking for owned user-controlled lines.

When the disconnect line adapter has been approved by NCP or has been forced by the MOSS operator, your code can take appropriate action on the associated lines if an entry point was specified on the UGLOBAL keyword of the GENEND definition statement.

The user parameter and status area acts as the interface when control is passed to your code for a disconnect-line-adapter function. It indicates the reason for passing control (disconnect line adapter) and provides any pertinent information for the function. The code at the entry points specified on the UGLOBAL keyword of the GENEND definition statement must route control to the appropriate user routine based on information passed in the user parameter and status area.

Connect Line Adapter

When a connect-line-adapter request is made from the MOSS console, your code can take appropriate action on the lines attached to the adapter being connected if an entry point was specified on the UGLOBAL keyword of the GENEND definition statement.

The user parameter and status area acts as the interface when control is passed to your code for a connect-line-adapter function. It indicates the reason for passing control (connect line adapter) and provides any pertinent information for the function. The code at the entry points specified on the UGLOBAL keyword of the GENEND definition statement must route control to the appropriate user routine based on information passed in the user parameter and status area.

Two Input/Output Controller (IOC) Buses

There are two IOC buses on the IBM 3745. Be sure that each IOH instruction that is issued is issued to the correct bus. If you use channel adapter subchannels, you need to combine the channel adapter register X'0F' image saved in external register 50 with the bus indication from external register 77 (saved in external register 53) to determine which channel adapter is raising the interrupt in level 3.

Note: The IBM 3745-130, 3745-150, 3745-160, 3745-170, and 3745-17A Communication Controllers have only one IOC bus.

16 Channel Adapters

Your code needs to handle up to 16 channel adapters. To uniquely identify a channel adapter, use a sequential number from 0–F, rather than just the three select bits, when interfacing with NCP.

When you want your code to modify channel adapter fields in certain control blocks using the NEOCBCH macro, your code must pass the channel adapter position (select bits). On the IBM 3745, your code must pass the sequential number of the channel adapter (0–F), rather than just the GGS bits.

Type 6 Channel Adapter

This section explains changes to your code required by type 6 channel adapters.

Ground Fault Errors: Your channel code is given control by level 1 if a ground fault error is detected, if the ground fault error counter is not exhausted, and if a level 3 interrupt is pending on a user ESC. If your code is given control during the disable processing as a result of a permanent ground fault error, it should not attempt to send status.

I/O Error Alerts: Your channel code must use the facilities of the line timer if you wish to run backup timers for input or output error alerts that your code issues on its ESCs.

Channel Adapter Logic Checks and Microcode Errors: Your code gets control for channel adapter logic checks and microcode errors in the same way it gets control with the IBM 3720 for internal adapter errors.

Bus Switching

Bus switching is the ability to switch the IOC buses from one CCU in an IBM 3745 Communication Controller to the other CCU. The two different functions of bus switching are fallback and switchback.

Note: The IBM 3745-410, 3745-41A, 3745-610, and 3745-61A Communication Controllers support fallback and switchback operations.

Fallback: A fallback operation in an IBM 3745 allows one CCU to overtake the IOC buses from the other CCU. This bus switch happens automatically when a CCU hardware failure occurs, or the operator may cause it manually from the MOSS console.

If the IOC buses are physically switched to an operational CCU (fallback), all of the resources attached to the buses are physically switched. The NCP running in the "switched-to" CCU logically acquires the switched resources, using a mini-initialization approach (on the switched resources). When a fallback occurs, your code gets control if you have coded an entry point on the UGLOBAL keyword of the GENEND definition statement. During the fallback mini-initialization, your code gets control in the following ways:

- When ports are logically added to a switched-line adapter, if one of the ports is a user-defined port
- When a switched-line adapter is logically connected, if one of the lines on the adapter is a user-defined line

- When a switched user-defined channel adapter is logically added (in this case your code must set AITSTAT bit 2 to B'1' in this channel adapter's AIT entry to indicate that the channel adapter is user-written and active)
- When a switched user-defined channel adapter is logically connected
- After all switched resources have been processed, if any switched resources are user-defined.

Also during the fallback operation, NCP quiesces each NCP line and SSCP-compatible (COMPOWN=YES) user-defined line, so ACTLINKs or DACTLINKs can be queued until the fallback operation is complete.

Switchback: A switchback operation in an IBM 3745 allows one CCU to return the IOC buses that were switched during fallback to the other CCU. The two types of switchback are conditional and unconditional.

Conditional switchback is a two-stage process. First, NCP checks all NCP lines, SSCP-compatible user-defined lines (COMPOWN=YES), and all channels to see if they are owned or active. If all resources are unowned and inactive, the second stage is executed; otherwise, the switchback is terminated. Second, NCP logically switches the previously switched resources away and requests MOSS to switch the buses physically. This processing is identical in both conditional and unconditional switchback and is referred to as the *common switchback code.*

An unconditional switchback does not check for active or owned resources before switching the buses back to the other CCU. Only the MOSS operator can invoke switchback from the console, specifying a conditional or an unconditional switchback.

To begin the common switchback code, you initiate Force Deactivate for all NCP lines and SSCP-compatible user-defined lines (COMPOWN=YES), and use the auto network shutdown function on all channels. For lines that are SSCP-compatible, you must reset LNVT bit 5 to B'0' for that line during your XIO IMMEDIATE processing (due to the Force Deactivate). After the lines have been deactivated and the channels have been shut down, the common switchback code logically switches the switched resources away. Your code gets control if you have coded an entry point on the UGLOBAL keyword of the GENEND definition statement. Your entry point gets control in the following situations:

- When ports are logically deleted from a switched-line adapter, if one of the ports is a user-defined port
- When a switched-line adapter is logically disconnected, if one of the lines on the adapter is a user-defined line
- When a switched user-defined channel adapter is logically deleted
- When a switched user-defined channel adapter is logically disconnected
- After all switched resources have been processed, if any switched resources are user-defined.

If an entry point is coded on the UGLOBAL keyword of the GENEND definition statement, the user parameter and status area acts as the interface when your entry point receives control. Your entry point should be a router routine that examines the operation field of the user parameter and status area and routes control to the appropriate user routine.

Channel Adapter Concurrent Maintenance

When a channel adapter concurrent maintenance (CACM) request is invoked, your code is given control if an entry point was specified on the UGLOBAL keyword of the GENEND definition statement.

The new user parameter and status area acts as the interface when control is passed to your code during a CACM operation. This parameter and status area indicates the reason for passing control (disconnect channel-adapter-in-use check, initiate disconnect channel adapter, disconnect channel adapter complete, connect channel adapter initialize, and so on) and provides any pertinent information for the function. The code at the entry point specified on the UGLOBAL keyword of the GENEND definition statement must route control to the appropriate user routine based on information passed in the user parameter and status area.

For all CACM operations your code will have the option to update its control blocks to reflect changes in the state of the channel adapter as the CACM operation proceeds.

For all CACM operations except disconnect channel-adapter-in-use check, a status of X'00' should be returned in the parameter and status area.

In the case of the CACM disconnect channel-adapter-in-use check, the code will have the opportunity to inform NCP whether it is using the channel adapter to be disconnected from the CACM mode. Your code should return a status of X'10' in the parameter and status area if it has any operation in progress on the channel adapter to be disconnected; otherwise, a status of X'00' should be returned.

Generating a Customized Controller Load Module

When adding routines to NCP, you assemble those routines, code macros to generate custom control blocks, and code linkage-editor statements to include your assemblies with standard NCP code. You must also code definition statements and keywords required for custom line control and programmed resources. See "Statements and Keywords to Generate a Customized Controller Load Module" on page 42 for the specific statements and keywords to code.

Note: The executable portion of an NCP load module cannot refer to labels above 4MB with LA, BAL, and BLG instructions because of the addressability limits of the IBM 3745 Communication Controller instruction set. NCP executable code must reside below the 4MB boundary.

The following sections describe the procedures for adding your controller code to NCP.

Including User-Written Code in NCP

See *NCP, SSP, and EP Generation and Loading Guide* for information on coding job control language (JCL) and setting up libraries for generation.

Assemble your controller code prior to generating any NCPs that will include the logic. MVS and VM use the CWAX assembler; VSE uses the IFZASM assembler. NCP includes this code through the use of INCLUDE and ORDER statements in the NCP link-edit. You must code your generation application to pass the appropriate INCLUDE and ORDER statements to NDF.

The control block macros must reside in libraries that are concatenated on the SYSLIB chain. Under VM, these libraries must be concatenated on the SYSLIB chain using a FILEDEF command and must be defined using the GLOBAL MACLIB command. Under VSE, these libraries must be included on a LIBDEF source statement. Control blocks can be specified in this copy code by calls to macros.

Under MVS (to be consistent with NCP), make sure that the text for the program logic is link-edited into a load module library before generating any NCPs that will include this code. A data definition (DD) statement for this library must appear in the job step for the NCP link-edit. The ddname on this statement must match the ddname coded on the INCLUDE statements.

To add to your code using user-written generation applications, do the following:

Step 1. Assemble your routines and place the resulting modules in the appropriate load library.

MVS Place the object modules in a library. Define this library by a data definition statement in the link-edit step.

Note: The ddname you use must match the ddname on the linkage-editor INCLUDE statements that you use for this member.

VM Place the object modules in a TXTLIB, which you must define by a FILEDEF command before the linkage-editor is called.

VSE Catalog the object modules as members of a VSE library, which you define by the OBJ LIBDEF statement during the link-edit step.

Step 2. Determine which of the following categories is appropriate for each module:

1. Level 2 and level 3 code that must reside in the first 64KB (KB equals 1024 bytes) of controller storage. The routines coded on the level 2 and level 3 keywords of the GROUP definition statement are the only routines that should be in this category. Because of space limitations, put a single branch instruction in the low 64KB of controller storage, and place the actual code in storage above 64KB. The address of the single branch instruction is used as the address of the code in halfword fields of control blocks such as GCBL2 and GCBL3. When control is passed to the address, the single branch in the first 64KB of storage branches to the actual code.
2. Level 2 and level 3 code that can reside anywhere in controller storage. The routines referred to by keywords other than LEVEL2 and LEVEL3 of the GROUP definition statement in the generation definition should reside in storage above 64KB.
3. Level 5 code that should *always* be inserted above 64KB.
4. Initialization code that is to be overlaid by the buffer pool when initialization has been completed.
5. User code that is to reside in read-only storage above 64KB.

Step 3. Write generation applications that process the statements and keywords necessary for your custom code. See *NCP, SSP, and EP Generation and Loading Guide* for information on writing generation applications. The applications must also pass source statements to NCP assembly to create user

control blocks and create link-edit control statements that include and order your custom modules and control blocks.

For CSECT names, your application must create INCLUDE and ORDER statements as follows for each operating system.

MVS Code each CSECT name on an ORDER statement, following the categories determined in Step 2.

CSECTs for which there is no ORDER statement are placed at the end of the load module. The buffer pool overlays the CSECT when initialization is complete.

VM Code each CSECT name on an ORDER statement, following the categories determined in Step 2.

CSECTs for which there is no ORDER statement are placed at the end of the load module. The buffer pool overlays the CSECT when initialization is complete.

VSE There are no VSE ORDER statements; create only INCLUDE statements.

Step 4. Use the ICNLEPTN NDF utility to post the INCLUDE and ORDER statements to the proper cluster for each category, as defined in Step 2. The clusters roughly correspond to the keywords used when generating NCPs using the GENEND definition statement. Table 5 shows how to specify these clusters.

Table 5. Cluster Categories for User-Written Generation Applications

Category (from Step 2)	MVS and VM	VSE
2a	Cluster 3	Cluster 3
2b	Cluster 4	Cluster 4
2c	Cluster 2	Cluster 2
2d	Cluster 7 and either cluster 2 or cluster 5	Cluster 7 and either cluster 2 or cluster 5
2e	Cluster 9	Cluster 9

Step 5. Assemble or compile your applications to produce executable user generation load modules.

Step 6. Place the user generation load modules in the appropriate library.

MVS Place the object modules in a library. Define this library by a data definition statement in the link-edit step.

Note: The name that you use must match the name on the linkage-editor INCLUDE statements that you use for this member.

VM Place the object modules in a TXTLIB, which you must define by a FILEDEF command before the linkage-editor is called.

VSE Catalog the object modules as members of a VSE library which you must define by the OBJ LIBDEF statement during the link-edit step.

Statements and Keywords to Generate a Customized Controller Load Module

You may include special device support and functions in your NCP through special IBM products such as Network Terminal Option (NTO), Network Routing Facility (NRF), and X.25 NCP Packet Switching Interface (NPSI), or user-written code. *NCP and SSP Customization Reference* for information on the support and functions of special IBM products and on writing and including user-written code in NCP. See the installation manuals of the IBM products you use for detailed information on how to define these programs and how to modify NCP to include them.

The following sections summarize the changes you must make to your generation definition in order to use customized resources and functions in your NCP configuration. See *NCP, SSP, and EP Resource Definition Guide* and *NCP, SSP, and EP Resource Definition Reference* for more information about using the definition statements and keywords mentioned.

Defining Resources Supported by User-Written Code

You can define the following types of functions and resources that can be managed by user-written code in your generation definition:

- Line control functions
- Programmed resources
- Channel control functions.

Defining User-Written Line Control: When defining line control functions managed by user-written code, you can use the following keywords to indicate the type of support provided.

```
GROUP BERPROC
      COMPACB
      COMPOWN
      COMPSWP
      COMPTAD
      LEVEL2
      LEVEL3
      LEVEL5
      LINEADD
      LINEAUT
      TIMER
      USERID
      XIO

LINE  AUTUACB
      UACB
```

Defining User-Written Programmed Resources: When defining programmed resources managed by user-written code, you can use the following keywords to indicate the type of support provided.

```
GROUP VIRTUAL

LINE  LINECB
      LINEFVT

PU    PUCB
      PUFVT
      PUNTFY

LU    LUCB
      LUFVT
      LUNTFY
      NUMSESS
```

Defining User-Written Channel Control: When defining channel control functions managed by user-written code, you can use the following keywords to indicate the type of support provided.

```
BUILD UCHAN

GROUP CAEXIT
      CHANLNK

LINE  ADDRESS

LU    UCCB
```

Including User-Written Code in the NCP Load Module

When you use user-written code, you must include it in the NCP load module. The following sections describe ways you can do this using definition statements and keywords.

Specifying User-Written Code Entry Points: The following keywords may be used to specify entry points to user-written code for special functions.

```
GENEND INIT
       TMRTICK
       UACCTNG
       UGLOBAL
```

For example, if your user-written code includes the routine USRL5C1, which functions as a timer-tick routine, specify `TMRTICK=(USRL5C1,...)`.

Including User-Defined Control Blocks and Tables: The following keywords may be used to specify assembler source code for user-defined control blocks and tables assembled with NCP control blocks and tables.

```
GENEND SRCHI
       SRCLO
```

For example, if your timer-tick routine requires that the user-defined control block USERTCB be included in the NCP load module and if your user macro library includes a member, USERTCB, which defines this control block, then specify `SRCHI=(USERTCB,...)` to include the DSECT defining USERTCB.

Specifying CSECT Names For User-Written Code: The following keywords may be used to specify the position in the NCP load module of CSECTs for user-written code using link-edit ORDER statements.

```
GENEND KEY0ORD
       ORDHI
       ORDINIT
       ORDLO
       ORDL2HI
       ORDL2LO
```

For example, if your user object library contains the object module USRL501, which contains the CSECTs USRL5C1 and USRL5C2, and your user macro library contains the member USERL2O, which includes the link-edit statement `ORDER USRL5C1,USRL5C2`, you can specify that the CSECTs USRL5C1 and USRL5C2 be located in the NCP load module below the 64KB address boundary. You can do this by coding ORDL2LO=(USERL2O) on the GENEND definition statement.

Including User-Written Object Modules: Macro library members may be specified on the following keywords to indicate where in the NCP load module the user-written code is to reside. The member named must use the link-edit INCLUDE statements to specify object modules that are members of a user object library.

```
GENEND KEY0INC
       INCHI
       INCINIT
       INCLO
       INCL2HI
       INCL2LO
```

For example, if your user object library contains the object module USRL501, and your user macro library contains the member USERL2I, which includes the link-edit statement `INCLUDE USERLIB(USRL501)`, you can specify that the object module USRL501 be included in the NCP load module below the 64KB address boundary. You can do this by coding INCL2LO=(USERL2I) on the GENEND definition statement.

Including User Routines in NCP Using the GENEND Definition Statement

See *NCP, SSP, and EP Generation and Loading Guide* for information on coding JCL and setting up libraries for generation.

You must first supply table 1 assemblies that define your control blocks to NDF. Place the table 1 assembly source code in a member of a source library and specify the member name on the SRCLO or SRCHI keyword of the GENEND definition statement.

These blocks are copied into an NCP table assembly during the second phase of NDF execution. In the MVS and VM environments, NDF dynamically invokes the IHR assembler for the second phase. Under VSE, a separate job step executing the IFZASM assembler handles the second phase.

Under MVS, concatenate this source library on the SYSLIB chain with the macro libraries using the JCL for the NDF run. In this case, the control block macros must also reside in libraries that are concatenated on the SYSLIB chain.

Under VM, concatenate this source library on the SYSLIB chain using a FILEDEF command. This source library must also be defined using the GLOBAL MACLIB command.

Under VSE, include this source library on a LIBDEF source statement. Control blocks can be specified in this copy code by calls to macros.

Using the CWAX assembler, assemble your controller code prior to the generation of any NCPs in which the logic will be included. NCP includes this code using the INCLUDE and ORDER statements in the NCP link-edit. You must create members of a source library that contain the appropriate INCLUDE statements and members that contain the appropriate ORDER linkage-editor statements. Specify the names of these members on the appropriate INC and ORD keywords on the GENEND definition statement. Concatenate the library on the SYSLIB chain. See *NCP, SSP, and EP Resource Definition Guide* for more information on coding the GENEND definition statement.

Under MVS (to be consistent with NCP), ensure that you link-edit the text for the program logic into a load module library. Before generating any NCPs that include user-written code, include a DD statement for this library before the job step for the NCP link-edit. The ddname on this statement must match the ddname coded on the INCLUDE statements.

The following steps explain how to add your code to NCP using the GENEND definition statement.

Step 1. Assemble your routines and place the resulting modules in the appropriate load library.

MVS Place the object modules in a library defined by a data definition statement in the link-edit step.

Note: The ddname that you use must match the ddname on the linkage-editor INCLUDE statements that you use for this member.

VM Place the object modules in TXTLIB, which you must define by a FILEDEF command before the linkage-editor is called.

VSE Catalog the object modules as members of a VSE library.

Step 2. Determine which of the following categories is appropriate for each module:

1. Level 2 and level 3 code that must reside in the first 64KB of controller storage. The routines coded on the level 2 and level 3 keywords of the GROUP definition statement are the only routines that should be in this category. Because of space limitations, put a single branch instruction in the low 64KB of controller storage and place the actual code in storage above 64KB. The address of the single branch instruction is used as the address of the code in halfword fields of control blocks such as GCBL2 and GCBL3. When control is passed to the address, the single branch in the first 64KB of storage branches to the actual code.
2. Level 2 and level 3 code that can reside anywhere in controller storage. The routines referred to by keywords other than LEVEL2 and LEVEL3 of the GROUP definition statement in the generation definition should reside in storage above 64KB.

3. Level 5 code that should *always* be inserted above 64KB.
4. Initialization code that is to be overlaid by the buffer pool when initialization has been completed.
5. User code that is to reside in read-only storage above 64KB.

Step 3. Within each of the categories determined in Step 2, code the name of each module on a linkage-editor INCLUDE statement. Place all the INCLUDE statements in a library. You can place all of the INCLUDE statements within one category in one member of the library, or you can further categorize the modules and create a separate member of the library for each of the additional categories.

MVS Code each CSECT name on a linkage-editor ORDER statement, following the categories determined in Step 2. Place the ORDER statements in the library. As with the INCLUDE statements, you can place all the macro statements within one of the previously named categories in one member of the library, or you can further categorize the modules and create a separate member for each additional category. The first phase of the generation process reads the linkage-editor control statements (INCLUDEs and ORDERs), and writes them into the linkage-editor control statement data stream.

If there is an INCLUDE statement for a module that contains a CSECT for which there is no ORDER statement, the linkage-editor places the CSECT at the end of the load module. The buffer pool overlays the CSECT when initialization is complete.

Define the library containing the linkage-editor control statements by a JCL data definition statement as a concatenated data set to the SYSLIB data definition statement in phase 1 of the generation job.

VM Code each CSECT name on a linkage-editor ORDER statement, following the categories determined in Step 2. Place the ORDER statements (one for each CSECT) in the library. As with the INCLUDE statements, you can place all the macro statements within one of the previously named categories in one member of the library, or you can further categorize the modules and create a separate member for each additional category. The first phase of the generation process reads the linkage-editor control statements (INCLUDEs and ORDERs) and writes them into the linkage-editor control statement data stream.

If there is an INCLUDE statement for a module that contains a CSECT for which there is no ORDER statement, the linkage-editor places the CSECT at the end of the load module. The buffer pool overlays the CSECT when initialization is complete.

Define the MACLIB containing the linkage-editor control statements using a FILEDEF command concatenated to the SYSLIB FILEDEF command in phase 1 of the generation job. Also define the SYSLIB using the GLOBAL MACLIB command.

VSE Provide the members of the library that contain the INCLUDE statements as input to the linkage-editor by coding the keywords of the GENEND definition statement (shown in Step 4 of this procedure) with the appropriate user member names.

Note: Because INCLUDE and ORDER statements are stored as 80-character records, you can ensure optimum use of storage by grouping as many CSECT names as possible on each INCLUDE and ORDER statement. The entire statement must not exceed 80 characters.

Step 4. Table 6 shows how to code the names of the members created in the previous steps on the appropriate GENEND keywords.

Table 6. Cluster Categories for GENEND Definition Statement

Category (from Step 2)	MVS and VM	VSE
2a	INCL2LO, ORDL2LO	INCL2LO
2b	INCL2HI, ORDL2HI	INCL2HI
2c	INCLO, ORDLO	INCLO
2d	INCINIT, either INCHI or INCLO, and ORDINIT	INCINIT and either INCHI or INCLO
2e	KEY0INC, KEY0ORD	KEY0INC

Table 7 shows how keywords specified on the GENEND definition statement affect the control block move.

Table 7 (Page 1 of 2). Definitions of GENEND Keywords

GENEND Keyword	Definition and Boundary Restrictions	DMA Key	Protect Key	Label ID
SRCHI	Source code for user-defined control blocks and tables that can reside above 4MB	R	1	$NEOMHI
INCHI	Specifies library members that contain INCLUDEs for non-L2/L3 modules that can reside above 4MB	R	1	$NEOMHI
ORDHI	Specifies library members that contain ORDERs for non-L2/L3 code that can reside above 4MB	R	1	$NEOMHI
SRCLO	Same as SRCHI except that it resides between 64KB and 4MB	R	1	CXALIGN3
INCLO	Same as INCHI except that it resides between 64KB and 4MB	R	1	CXALIGN3
ORDLO	Same as ORDHI except that it resides between 64KB and 4MB	R	1	CXALIGN3
INCL2HI	Specifies library members that contain INCLUDEs for L2 and L3 modules that reside between 64KB and 4MB	R	0	CXALIGN2
ORDL2HI	Specifies library members that contain ORDERs for L2 and L3 modules that reside between 64KB and 4MB	R	0	CXALIGN2

Table 7 (Page 2 of 2). Definitions of GENEND Keywords

GENEND Keyword	Definition and Boundary Restrictions	DMA Key	Protect Key	Label ID
INCL2LO	Same as INCL2HI except that it resides below 64KB	R	0	N/A
ORDL2LO	Same as ORDL2HI except that it resides below 64KB	R	0	N/A
INIT	Specifies entry points of user-written initialization routines that reside below 4MB	R	2	$NEOINIT
INCINIT	Specifies library members that contain INCLUDEs for user-written initialization routines	R	2	$NEOINIT
ORDINIT	Specifies library members that contain ORDERs for user-written initialization code	R	2	$NEOINIT
KEY0INC	Specifies library members that contain INCLUDEs for read-only storage modules that can reside below 4MB	R	0	CXALIGN4
KEY00RD	Same as KEY0INC for ORDERs	R	0	CXALIGN4
TMRTICK	Specifies the entry points of user-written timer-tick service routines; must reside in INCL2LO or INCL2HI	R	0	N/A
UGLOBAL	Specifies the entry points for user-written global routers	N/A	0	N/A
UACCTNG	Specifies the entry points of user-written accounting notification routines; resides in INCL2HI	R	0	CXALIGN2

Figure 10 on page 49 represents a map of NCP storage produced by an MVS link-edit; VSE and VM maps will be similar to this one. The names of the various parts of the map are taken from the GENEND keywords that specify the names of your routines.

Note: To determine how much storage you need in your communication controller for NCP buffers, see *NCP, SSP, and EP Generation and Loading Guide.*

IBM special products or user-written code must not place code into cluster 6 until NDF processes the BUILD definition statement. The $NEOMHI label must be at the start of cluster 6.

Location	Contents	Storage Protect Key	Read Only
000000			
	Lower 2KB of executable code	0	Y
2KB	Line Vector table (LNVT)	2	N
8KB	NCP code below 64KB	0	N
	Preassembled user code that must run key 0 below 128KB	0	N
CXALIGN1	NCP control blocks below 64KB	2	N
64KB	NCP control blocks above 64KB	2	N
CXALIGN2	User key 0 code above 64KB	0	N
	NCP executable code above 64KB	0	N
CXALIGN3	User code that runs in key 1 (INCLO/ORDLO)	1	N
	User block handler routines	1	N
	SRCLO user control blocks	1	N
CXALIGN4	User key 0 code (read only)	0	Y
	NCP read only area (not used)	0	Y
CXAL5SA	Level 5 save areas	7	N
CXSTART	CXFINITC	2	N
	NCP control blocks above 4MB	2	N
$NEOMHI	User written code INCHI/ORDHI/SRCHI control modules above 4MB	1	N
$BUFPOOL	Buffer pool	7	N
	Control block pool	2	N
UTILSIZE	Installed but unused storage	2	Y
	MOSS mailbox, LA workspace	2	N
TRUESIZE			

Figure 10. IBM 3745 Communication Controller NCP Storage Map

NDF Standard Attachment Facility

The NDF standard attachment facility (SAF) allows generation applications to process definition statements, keywords, and linkage editor control statements to customize the generation process. You can write your own generation application to generate user-written line control and define program resources, such as NetView Performance Monitor (NPM) and X.25 NCP Packet Switching Interface (NPSI). See Chapter 4 in this book and *NCP, SSP, and EP Generation and Loading Guide* for more information on the NDF standard attachment facility and generation applications.

Use the following keywords to include the NDF standard attachment facility in your generation process.

USERGEN on the OPTIONS definition statement specifies the names of the generation application load modules or IBM products that use the NDF standard attachment facility. Each generation application must have a separate generation application load module to process its definition statements and keywords and generate table source statements. You can specify a maximum of 25 generation load modules in a single generation.

LNKOWNER on the GROUP definition statement specifies the name of the user-written generation load module that does the link-level processing of resources defined on the GROUP definition statement. The name must match one of the user-written generation load module names specified for USERGEN on the OPTIONS definition statement.

VIROWNER on the GROUP definition statement specifies the name of the user-written generation load module that does the virtual processing of resources defined on the GROUP definition statement. The name must match one of the user-written generation load module names specified for USERGEN on the OPTIONS definition statement.

VIROWNER on the NCPNAU definition statement specifies the name of the user-written generation load module that owns this NCPNAU definition statement. The name must match one of the user-written generation load module names specified for USERGEN on the OPTIONS definition statement.

Chapter 2. Customizing NCP Line Control

NCP lets you provide line control support for devices and line protocols that are not normally supported by NCP. A typical example is providing support for non-SNA terminals that you would like to connect to your network. By providing your own line control for those terminals' lines, you can connect them to the SNA network. Your line control provides an SNA appearance to the rest of the network and may convert SNA commands to commands appropriate to your terminals. You can also customize channel support.

To provide line control, you can create routines that replace the normal NCP path control and scanner interface functions for your customized line group. You can also provide other functions, such as timer support, if you need them. For standard customized line control, you must supply the following routines:

- SNA command handler
- Level 2 interrupt handler
- Level 2 and level 3 routers
- Link scheduler, which includes:

  XIO handlers
  Link-scheduling routines
  Error handlers
  Command enders

- Timer routines.

These routines provide the minimum functions needed to customize line control. You may need to provide more functions than the routines listed provide. Your particular needs will determine which functions to provide. Use NCP and its functions as a guide in planning your customization effort.

Before you start to customize, you must have an excellent understanding of the NCP scanner operation, the protocol and hardware you are supporting, and the SNA PIUs and commands that your line control will be working with. You must also have a thorough knowledge of NCP operation. Refer to *Principles of Operation* for your controller for information on the scanner. Refer to *Systems Network Architecture Format and Protocol Reference* and to *Systems Network Architecture Formats* for detailed SNA information. Refer to *NCP and EP Reference* for information on NCP operation.

Control Blocks for User Line Control

NCP provides the user adapter control block (UACB) for your lines in place of the normal ACB and the user line vector table (ULVT), in addition to the line vector table (LNVT) used for NCP lines. NCP also provides a unique control block, the group control block (GCB).

Instead of the LNVT-PSA-ACB chain used for normal lines, NCP uses a LNVT-PSA-GCB chain for customized lines. The LNVT contains a pointer to a PSA for every line defined to NCP, including user-controlled lines. For each customized line, a ULVT entry points to the UACB for that line. All the PSAs for a customized

line group point to a single GCB, which is used instead of the ACB for initial addressing. Figure 11 shows this control block structure.

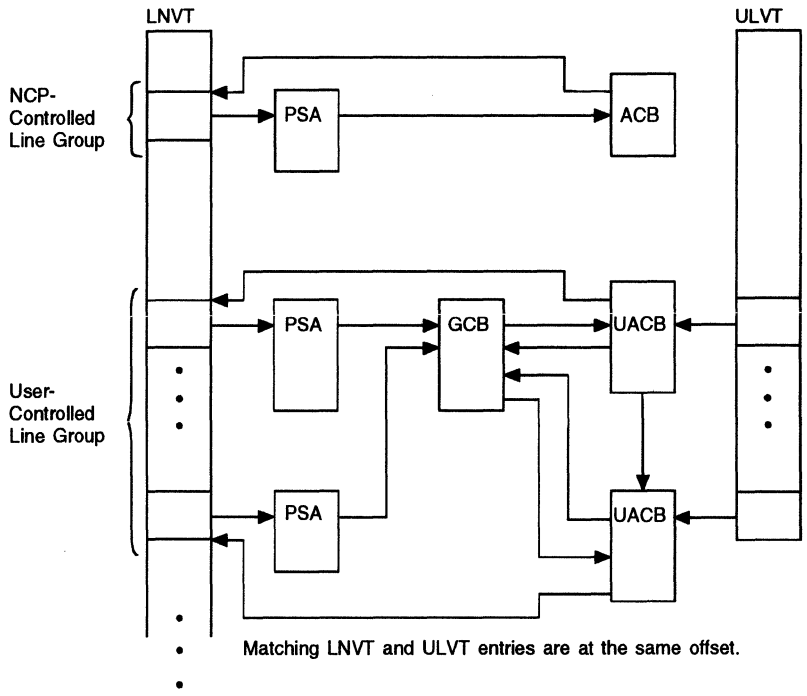


Figure 11. Control Block Structure for User-Controlled Lines

Group Control Block

One group control block (GCB) is defined for each group of user-defined lines and holds pointers to the routines for that group, which include the timer routines and XIO routines. The GCB also keeps the head and tail pointers for the UACB chains. The PSA for each line in the group and the UACB for each line point to the GCB.

Use the UACBGCBP field to find the GCB for a line.

User Adapter Control Block

The user adapter control block (UACB) provides the primary interface between your line and NCP. Each line's UACB points to the GCB for that line group and contains a pointer to the LNVT entry that points to the UACB. If you are providing your own level 5 services and are not using trace support, use the SDB generation macro to put the minimal required fields in your UACB. Table 8 on page 53 shows these fields. If you plan to use NCP level 5 services or provide trace support for your lines, you must provide the ACB fields listed in Table 9 on page 53 in your UACBs. Refer to *NCP and EP Reference Summary and Data Areas*, Volume 1, for specific offsets.

If you are using trace or level 5 services, then because NCP uses the UACB as if it were a normal NCP control block, you must specify a standard line protocol type in the CCBTYPE field even though your protocol is nonstandard. You can do this by

specifying the line type you want in the CCBTYPE field (SDLC, BSC, or SS) on the LNCT keyword of the GROUP definition statement.

Use the FINDUACB macro to find the UACB for a line.

Required Control Block Fields

The fields listed in Table 8 are required for line control routines. If your control blocks have only this level of compatibility, you must write your own level 5 routines to handle the functions that NCP would otherwise handle.

Table 8. UACB Fields Required for Line Control Routines

UACB Information	Corresponding CCB Field
Address of user level 2 interrupt handler	CCBL2
Address of user level 3 interrupt handler	CCBL3
Line type (1 byte)	CCBTYPE
Expected status indicator (must always be zero)	CCBESTAT
GCB flags (identification of control block as a GCB)	CCBSETYP
Addresses of user timer routines	none
Pointer to transmit leg of duplex link	CCBXACBP
Pointer to AXB/UAXB	CCBAXBP

Fields Required When COMPACB=YES

The control-block fields in Table 9 must be in your UACB if you code COMPACB=YES on the GROUP definition statement. These compatibility requirements are in addition to the requirements for IBM 3745 functions.

Table 9. UACB Fields Required when COMPACB=YES for BSC, Start-Stop, and SDLC

Field Description	BSC or Start-Stop	SDLC
I/O command field	IOBCMAND	LXBCMAND
Command modifiers field	IOBCMODS	LXBCMODS
Pointer to first buffer received	IOBDATAP	LXBDATAP
Receive leg status	IOBSTATR, IOBERST	LXBERST, LXBHSTAT
First error extended status	IOBEREST	LXBEREST
Extended error status	IOBEXTST	LXBEXTST
Pointer to last buffer in chain	IOBFNLPT	LXBFNLPT
Immediate control flags	IOBIMCTL	LXBIMCTL
Input control data address	IOBINPUT	LXBINPUT
Pointer to line control block	IOBLCB	LXBLKBP
Outcome of command operation	IOBSTAT	LXBSTAT
First data offset	IOBOFSET	(not applicable)

Table 10 lists CCB and AXB fields required when COMPACB=YES.

Table 10. CCB and AXB Fields Required when COMPACB=YES

Field Description	CCB or AXB
LPDA control byte	AXBLPDA
Pointer to the buffer used for receive operations	AXBLPDRB
Pointer to the buffer used for transmissions	AXBLPDXB
Command and line address in scanner	AXBR1
Scanner address and E bit	AXBR2
Pointer to the dial digits	AXBSDIAL
Indicates the active step of the modem dial process	AXBSTATE
Line type	AXBTYPE
Pointer to ACB extension (AXB)	CCBAXBP
Pointer to LNVT (backward pointer)	CCBBAR
Start-stop burst mode flags and character count	CCBCCNT
Buffer character count	CCBCHAR
Control flags or line type	CCBCTL
Buffer maximum for receive operation	CCBCUT
Address of data byte sent or received	CCBDATA
Ending line operational status	CCBEND1
Error retry counter	CCBERCNT (pre-SNA only)
Error retry limit	CCBERTRY
General flags	CCBFLAGS
Address of first buffer in a block	CCBHDBUF
Pointer to line group control block	CCBLGPT
Pointer to next UACB in levels 2 and 3 chain	CCBLINK
Buffer for next transmit character	CCBNEXT (pre-SNA only)
Pointer to receive leg of duplex link	CCBRACBP
Control flags	CCBRSPON
Setmode control flags	AXBSMSDF
Address of current buffer	CCBSTART
Current line operational status	CCBSTAT1
Pointer to next UACB in timer chain	CCBTACB
Time-out interface	CCBTIME
Time-out command	CCBTOCMD
Time-out remembrance	CCBTOREM
Timer work entry for this UACB	CCBTWORK
Dial control flags	CCBTYPEC
Transmit turnaround (LCD/PCF)	CCBXTPCF

Fields Required When COMPOWN=YES

Table 11 shows the fields required when COMPOWN=YES.

Table 11. UACB Fields Required when COMPOWN=YES

Field Description	BSC or Start-Stop	SDLC
Pointer to line or link control block	IOBLCB	LXBLKBP
SNP mask of SSCP that issued activate link	LCBSNPM	LKBSNPM
Link quiesce pending bit	LCBACTNS (bit 1)	LKBSTAT (bit 3)
Resource ID of the line	LCBRID	(not applicable)
	(not applicable)	LKBNWADR

Fields Required When COMPTAD=YES

Table 12 and Table 13 show the fields required when COMPTAD=YES.

Table 12. UACB Fields Required when COMPTAD=YES for BSC, Start-Stop, and SDLC

Field Description	BSC or Start-Stop	SDLC
Pointer to line or link control block	IOBLCB	LXBLKBP
Third line status byte	LCBLST3	(not applicable)
Miscellaneous flags	(not applicable)	

Table 13. CCB, AXB, or ACU Fields Required when COMPTAD=YES

Field Description	CCB, AXB, or ACU
Command and line address in scanner	AXBR1
Scanner address in E bit	AXBR2
Pointer to LNVT entry	CCBBAR
Pointer to ACB extension (AXB)	CCBAXBP, CCBSETYP
Dial control flags	CCBTYPEC
Autocall unit interface address	ACUBAR
First level retry count limit	ACUR1
Second level retry count limit	ACUR2

Fields Required When COMPSWP=YES

Table 14 and Table 15 show the fields required when COMPSWP=YES.

Table 14. UACB Fields Required when COMPSWP=YES for BSC, Start-Stop, and SDLC

Field Description	BSC or Start-Stop	SDLC
Pointer to link control block	IOBLCB	LXBLKBP
PEP flag field	IOBPFLAG	(not applicable)
Link status	LCBLSTAT, LCBLST2	LKBSTAT
AIT index field	LCBINDX	LKBINDX

Table 15. CCB, AXB, and ACU Fields Required when COMPSWP=YES

Field Description	CCB, AXB, or ACU
Command and line address in scanner	AXBR1
Scanner address in E bit	AXBR2
LPDA flags byte	AXBLPDA
Original relative line number	AXBOLDLN[1]
Current relative line number	AXBNEWLN
Dial original relative line number	AXBDODLN[2]
Dial current relative line number	AXBDNWLN[2]
Autocall unit interface address	ACUBAR
First level retry count limit	ACUR1
Second level retry count limit	ACUR2
Pointer to LNVT entry	CCBBAR
General flags	CCBFLAGS
Dial control flags	CCBTYPEC
Pointer to receive leg of duplex link	CCBRACBP
Pointer to ACB extension (AXB)	CCBAXBP

1 Must be initialized to the relative line number of the line.

2 Must be initialized to the dial port relative line number or, if there is no dial port, to zero.

User Line Vector Table

The user line vector table (ULVT) contains a pointer to the UACB for each customized line in NCP. The offset of each of these entries is the same as the offset for that line in the LNVT, so that a one-to-one correspondence exists between LNVT entries and ULVT entries. Locations in the ULVT that correspond to normal lines in the LNVT contain zeros, and the ULVT ends with the last customized line, even though the LNVT may be longer.

Writing User Line Control Routines

The routines discussed in this section make up the basic set of routines used in any custom line control. Your specific needs dictate what functions you need to provide; you may find you need to duplicate more NCP functions than are shown here. Use these routines as a guide in planning your customization effort.

This book cannot discuss all possible customization situations. However, this section gives you the conceptual basis for designing and implementing your own line control. For information about the attachments for user-written line control code in NCP, refer to the chapter on entrances and exits for user-written line control in *NCP and SSP Customization Reference*.

Your routines form the interface between the communication controller's scanner hardware and microcode and NCP's peripheral function. You use NCP physical services and intermediate node services as NCP does. Figure 12 shows where your code fits logically into NCP.

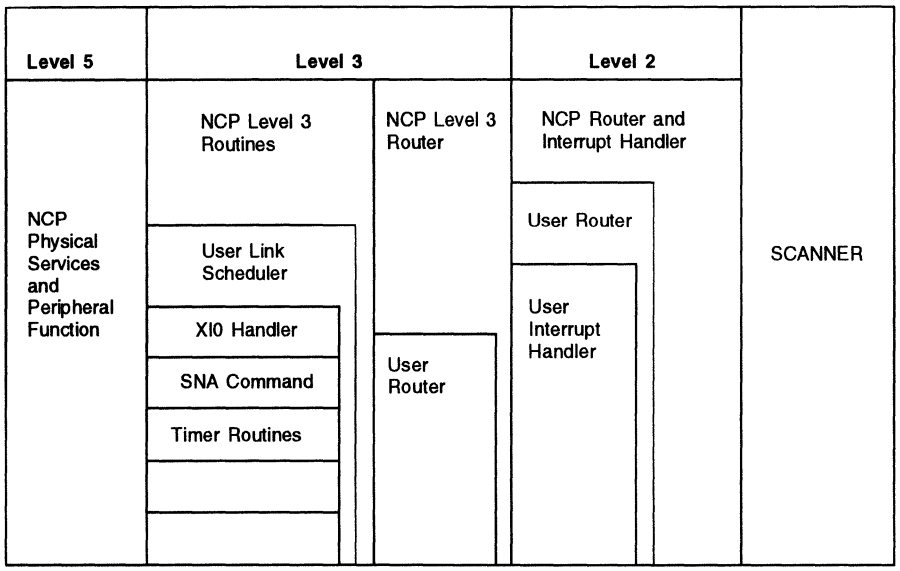


Figure 12. User Line Control in Relation to NCP

When you design your line control strategy, you should try to make your routines behave as much like normal NCP routines as possible. This will make your routines easier to design, write, test, and diagnose. Keeping your control block structure close to NCP structure makes it easier to maintain your code if and when NCP control blocks change. **The internal functioning of NCP can change at any time, so you must design ease of maintenance into your customized code.**

You must also minimize the path length of your level 2 routines. Level 2 operations are time-critical and your routines can severely affect NCP throughput. Your level 2 routines should use an average of 50 instructions and no more than 100 instructions.

If your routines generate error records, the records must be recognizable by the host processor's error analysis program. For the formats of the records for the con-

troller you are using, refer to the sections on RECMS RU formats in *NCP and EP Reference Summary and Data Areas*, Volume 2.

The following sections describe the basic routine types needed for custom line control.

SNA Command Handler

The SNA command handler provides an SNA appearance for your custom line control. It accepts SNA commands, converts them to commands appropriate for your lines, and returns any responses required by SNA.

Level 2 Router

Your level 2 router gets control from the NCP level 2 router whenever the scanner generates an interrupt for one of your customized lines. This router then calls the appropriate routine in your level 2 interrupt handler. The entry point for this routine is specified on the LEVEL2 keyword of the GROUP definition statement for each customized line group.

Level 2 Interrupt Handler

Your level 2 interrupt handler gets control from your level 2 router whenever the scanner requires service for your lines. Use the FINDUACB macro to get the UACB for the interrupting line. Use the POSTUACB macro to post the UACB to the CICP queue and issue an interrupt to level 3, where your level 3 routines will handle the line.

Level 2 F5 Command Processing

If you want to take advantage of the F5 command function (line dump) when the scanner backup timer expires, you can do the following:

- Put the address of NCP's F5 command level 2 processor (CXBL2A4) in the level 2 routine address field (CCBL2). This label is an entry for NCP; you must define it as an EXTRN.
- Write a level 2 F5 command cleanup routine and put its address in the AXBCBL2 field.
- Issue an F5 command. The suggested value is 6 seconds, although you can use a different value.

If a level 2 interrupt does not result from the F5 command (the 6-second timer expires), the normal NCP processing is invoked. This means that an 11B1 BER will be built (communication scanner processor command time-out) and a link INOP will be generated. The link will be disabled. You will not receive control unless you are doing your own timer processing. Any necessary F5 cleanup must be done when the link is reactivated.

If a level 2 F5 interrupt does occur, control will be passed to your level 2 F5 clean-up routine after NCP has executed its level 2 F5 routine CXBL2A4. CXBL2A4 builds an 11A4 BER (transient line error) and also builds an 11AB BER (integrated modem error), if appropriate. An EXIT should be taken when your F5 cleanup routine finishes; there is no need to return control to NCP.

Level 3 Router

Your level 3 router gets control when the NCP level 3 router determines that an interrupt is for one of your lines. These interrupts can be PCIs from level 4, timer interrupts, or other interrupting conditions. Your level 3 router then calls the appropriate level 3 routine. The entry point for this routine is coded on the LEVEL3 keyword of the GROUP definition statement for each customized line group.

Link Scheduler

Your link scheduler gets control from your level 3 router. In addition to the operations associated with link scheduling, such as polling lines and sending and receiving data, the link scheduler logically includes the XIO handling routines, the error handling routines, and the command ender.

The link scheduler is the heart of your customized code. It is responsible for all "work" operations for your lines, including contacting stations, polling lines, and doing the work necessary to send and receive data. The link scheduler must provide all the path control functions necessary for your lines.

Link-Metered Pacing

NCP uses link-metered pacing to prevent one session from dominating the use of a link. Each session may put only one PIU at a time on the link outbound queue (LOBQ). To prevent a second PIU from being enqueued on the LOBQ, the CPM-OUT QCB associated with the session is placed in the *not ready* state (that is, it cannot be dispatched again until the QCB is QPOSTed). Whenever a PIU is taken off the LOBQ, its QCB (which is stored in U2LUAQCB–X'10' in the PIU) must be QPOSTed. Failure to QPOST the QCB will result in a hung session until an expedited PIU is transmitted.

Link Session Priority

NCP uses link session priority to provide transmission priority on its peripheral links. Two things to keep in mind are:

- The transmission priority for the PIU is passed in the low-order byte of register 5.
- The U2LUAQCB field (offset X'10' in the PIU) now has multiple definitions depending on the value of bit 1 of ECBCSTAT (offset X'04' in the ECB). This bit indicates if the address in the field is the address of the QCB associated with the PIU (bit 1 is off) or the address of the last segment of a segment chain created by NCP (bit 1 is on).

Note: The new definition for the U2LUAQCB field is used only if you code LSPRI=LINK for the user line control group. Otherwise, the U2LUAQCB field will always be the address of the QCB associated with the PIU.

XIO Handler

Your XIO handler is the interface from NCP peripheral and physical services in level 5 to your link scheduler. NCP physical services control the activation, deactivation, and operation of your links. Peripheral services place PIUs on the link outbound queue. Level 5 issues one of four XIO macros to your link scheduler to request various link operations, such as the request to contact a station. Your XIO handlers must take the request issued from level 5 and convert it to the appropriate

command for your line protocol. Your link scheduler then carries out the actual function.

The XIO handler consists of four routines: XIO LINE, XIO SETMODE, XIO IMMEDIATE, and XIO LINK. XIO LINE, SETMODE, and IMMEDIATE form the physical services interface. XIO LINK interfaces with the peripheral services. XIO LINE, SETMODE, and IMMEDIATE run in level 3; XIO LINK runs in level 4. Code the entry points of these routines on the XIO keyword of the GROUP definition statement for each customized line group.

Timer Routines

Your timer routines provide time-out service for your lines. These services include handling time-out errors and doing shoulder-taps.

The NCP communication-line timer routine provides time-out services for NCP's telecommunication lines. Specify the entry points of your timer routines on the TIMER keyword of the GROUP definition statement.

In normal processing, the timer must use a minimal number of machine cycles. Normal processing includes:

- Detecting and accepting a new or changed timer command. Timer commands are specified by TVS macros.
- Decrementing the time value for a current timer command.
- Checking for certain error conditions.

The timer can use many machine cycles in processing actual time-outs and exceptions. For more information on the timer routines, refer to *NCP and EP Reference*.

In the timer routine, the original content of register 6 is the address of a save area; you must not change either the save area or its address in register 6. If you need a save area, establish a new one. Use the SAVE and RESTORE macros to manage save areas.

The timer must periodically exit and allow the level 3 interrupt with the highest priority to be given to NCP. In particular, the timer must not prevent the channel IOS routines from running as needed.

The timer executes in program level 3. Timer commands (TVS macros) can be issued in either level 2 or level 3. An interlock mechanism allows the timer routine to detect that a new timer command has been issued. This mechanism consists of a halfword field (CCBTIME) in the ACB, which has 1 byte that is read-only to the timer and 1 byte that is read-only to your routine (through the TVS macros).

The timer provides two levels of resolution: high-resolution service (0.1 second) is for BSC and SDLC lines; low-resolution service (0.5 second) is for start-stop lines. The type of service is determined during network generation. The NCP generation procedure uses the CCBTACB field of each ACB to chain all the high-resolution ACBs. The chain begins and ends in a section of the communication-line timer and RAS control table (CTB); the chain may be null (empty). The network generation program divides the low-resolution lines into five chains of ACBs and uses the CCBTACB field of each ACB to form the five chains of low-resolution ACBs. Not

all the chains have the same number of ACBs; some chains may in fact be null. Each of these chains begins and ends in a different section of the CTB.

UACBs are automatically placed in the high-resolution chain during network generation. The communication-line timer routine, CXCCLINT, gets control once each 100 milliseconds and may be reentered once for each time that control is surrendered. This allows the channel adapter interrupt handler to process level 3 interrupts for the channel adapter. The service cycle consists of processing the single high-resolution chain of ACBs and one of the five low-resolution chains each 100 milliseconds, regardless of whether a chain is null. During the processing, the timer suspends itself after processing a specified number of ACBs. This number is specified in the TIMLNCNT field of the byte direct addressable (XDB).

Each ACB in the chain of ACBs can have one of four timer conditions in effect. These conditions are assigned for each line during network generation and can be changed by issuing one of the TVS macros (TVSIDL, TVSMOD, TVSNEW, TVSREF, or TVSRTRN). One of the four timer conditions is always set; it is impossible to disable the timer. The timer conditions are as follows:

Idle: No action is to be taken when the current timer value expires. The timer value is a nonvariable system constant that shows that the line is not active.

Shoulder Tap: When a user-started timer steps to 0, control is passed to this entry point of the CXCCLINT routine. Level 2 interrupts are disabled. You are thus alerted to take action as appropriate and to restart the timer. NCP starts a timer of its own; if you take no action before the NCP timer times out, NCP assumes an error occurred, and passes control to the lagging shoulder-tap entry point. You set the timers by storing counts and codes in the timer-related areas of the ACB or the UACB. NCP determines whether you have taken any action by looking for a change in the stored code.

Lagging Shoulder Tap: A shoulder-tap time value has expired, and no new command has been given for the line within the allowed time (0.1 second for high-resolution and 2 seconds for low-resolution lines). NCP treats this as it treats an error time-out.

Error: Error time-outs are caused by hardware errors and by unserviced shoulder-tap time-outs. If the time value has expired, the timer routine issues a set of instructions that causes no operation by the hardware (except for switched lines), puts an idle routine in the CCBL2 field, queues the ACB on the CICP queue, and sets the CCBL3 field in the CCB.

In standard NCP, whenever further action is indicated, NCP changes the CCBL2 field in the ACB so that a routine other than the normal one gets control the next time an interrupt occurs for the line. With user-written line control, when the CXCCLINT module has determined which entry point should get control, the address of the entry point is obtained from the GCB. NCP passes control to your routine, which can carry out any processing necessary; you return to NCP by issuing the TVSRTRN macro. The UACB address is passed to your routine in register 5.

The following paragraphs describe NCP timer functions. For each line that is scanned by the timer module, one of the following occurs:

- The link activity time value is decreased by 1. If the time-out expires, the currently active command on the link is forced to end (by CXELNKCU), and no other timer function is carried out for that link.
- A new command is serviced. The CCBTIME field of the CCB is examined, and the interlock bits (the high-order bits of the CCBTIME field of the CCB) are compared. If the bits are not equal, a new command has been issued. The new command can be a reinitiation of the existing command, or a different command. In any case, the low-order byte (CCBTOREM) of the CCBTIME field is set equal to the high-order byte (CCBTOCMD) with all bits except the high-order bit set to zero.
- An old command is serviced. If the high-order bits of the 2 bytes of the CCBTIME field are alike, the command is old, the count is decremented, and appropriate action is taken when the time-out occurs. The command is then rechecked to ensure that a new command has not been issued since the count was decremented.

When a shoulder-tap time-out occurs, the time field in CTBWORK is set for one timer interrupt. The CCBL2 field is set to the address to which control is to be given at the next level 2 interrupt for the line.

When an error time-out occurs, the timer and RAS skip-action bits are set to one and a maximum timer-interrupt count is set.

In each of the actions described here, a test is made to determine if the line being serviced has user line control. The UACBGCBP field of the UACB contains the address of the GCB, and the GCB contains the address of the appropriate entry point to your timer service routine. Control is immediately passed to that entry point; you must return control by issuing a TVSRTRN macro.

Initialization Routines

You can provide initialization routines to set up any specialized control blocks you create to support your customized code.

Building Your Own RECMS/NMVT

You can provide routines that build RECMS and network management vector transport (NMVT) records. Your RECMS/NMVT build module can be called either by your own code or by NCP in place of its own RECMS/NMVT build routine. One advantage of letting NCP call your build module is that your RECMS or NMVT will be synchronized with other NCP processing (INOP, for example). If you choose to have NCP call your RECMS/NMVT build module, specify the label for the RECMS/NMVT routine entry point on the USERID keyword of the GROUP definition statement and on the INCL2HI keyword of the GENEND definition statement.

Your RECMS/NMVT build module must use the SUBRTN macro to link to NCP code. You must code the RECMS/NMVT build module to accept a pointer to the LKB in register 2. Your routine must provide its own save area, if one is required. NCP will do a PRELEASE prior to calling your module; however, your code must do the LEASE. If your code cannot get sufficient buffers to build the RECMS or

NMVT, your code should release the buffers and return control to NCP. Your code must route the RECMS or NMVT to its proper destination.

Building Generic Alerts

Depending on the modem in use, generic alerts can replace RECMS as the error-reporting tool for dial-out failures. Generic alerts are a type of major vector or NMVT RU. Because the exact cause of a dial failure may not be known, generic alerts normally document several possible causes as well as status information returned from the scanner.

The PRODID keyword lets IBM special products or user-written code include product identifier information in the product set ID subvector of the generic alert. NDF accepts the PRODID keyword only for IBM special products or user-written code lines. If you do not want to include this product information, do not code the PRODID keyword.

PRODID=label is coded on the GROUP definition statement. The label you code should be an address label for an area in storage that contains the IBM special products or user-written code product identifier (subvector X'11'). The subvector X'11' you define should be less than X'80' bytes in length.

If you want IBM special products or user-written code to build the generic alerts, code USERID=(,,NORECMS) on the GROUP definition statement. This enables IBM special products or user-written code to build the entire generic alert. User-written code can then also choose to invoke the GALERT macro to build the common portions of the generic alert. These common portions are:

- Transmission header (TH)
- Request/response header (RH)
- NMVT request/response unit (RU) header
- Generic alert length bytes and major vector code
- SNA address list subvector X'04'
- Product set ID X'10' MS subvector
- IBM Special Products or User-Written Code product set ID X'11' MS subvector (included only when the PRODID keyword is coded)
- Relative time X'42' MS subvector.

To use the GALERT macro, you must specify enough buffers in your code to hold the common generic alert information. This includes enough buffers to hold bytes required by NCP and the IBM special products or user-written code product identification subvector. You must compute how many buffers the generic alert needs based on NCP buffer size. Once the GALERT macro is executed, the buffers will be returned to IBM special products or user-written code with valid data counts and offsets filled in. If you define excess buffers, the buffers are returned exactly as they were received. If you specify an inadequate number of buffers, you receive a negative return code.

Supporting Diagnostic Aids

SSP provides a line trace and a scanner interface trace (SIT) for binary synchronous (BSC), start-stop (SS), and synchronous data link control (SDLC) lines. Wrap tests are also available through the controller itself. You can provide support for these diagnostic aids for a group of customized lines by making the UACB for that group closely compatible with the ACB, as shown in Table 9 on page 53, and by coding COMPACB=YES on the GROUP definition statement for that group. You can also use the FETRACE macro to do supervisor and dispatcher tracing.

See the *NCP, SSP, and EP Diagnosis Guide* and the *NCP, SSP, and EP Trace Analysis Handbook* for more information on the SIT and line trace. Refer to *3745 Communication Controller Problem Determination and Extended Services* and *3720 Communication Controller Problem Determination and Extended Services* for more information on wrap tests.

When NCP is doing the traces or wrap tests, your code does not get control.

Handling XID Exchange

Your line control code must handle the XID exchange for all SDLC-like switched lines and for SDLC-like nonswitched lines for which XID=YES is coded on the PU definition statement. The amount of processing your code must perform may vary. As a minimum, your code may need to pass the XID PIUs between NCP and the physical units you are supporting. Where greater levels of processing are required, for example, your line control code may need to set XID3 fields that would normally be set by the physical unit sending the XID, or it may even be necessary to create the entire response XID. As another example, only the NCP physical unit can initiate XID exchanges over an already active session. If the physical unit for which you are providing line control is capable of initiating an XID exchange, your code must intercept those PIUs, because NCP is not able to accept them.

Your XID handler resides in level 3 and interfaces with NCP's XID handler in level 5.

The XID given to NCP must be at least 6 bytes long; shorter XID PIUs should be rejected by your line control and retried. The actual length of the XID must match the length value given in the length field of the XID.

The first 29 bytes of the XID must be in the first buffer and cannot start in the first 19 bytes. XIDs built by level 5 always start at byte 20.

When the RUN command ends, level 5 assumes that an XID has been received in response to the XID it sent. Note that the buffers containing the transmitted XID are not released, but are returned to level 5. The LXBDATAP field of the LXB must contain a pointer to the XID sent. The LXBINPUT field points to the XID received in response. Even in the case of station or link error, the XIDs should be returned to level 5, rather than being released in level 3.

The station address must be at byte X'13' and is 1 byte long.

If the resource using this line is represented by a GCB, bit 1 of the format 3 XID field XID3LSCP (ABM capability) is not checked.

Refer to *NCP and EP Reference* for more information on XID exchanges.

Using NPM Data Collection

You can use the NetView Performance Monitor (NPM) performance management function with NCP to accumulate network data about user-written line control resources. The NPM function collects the data, logs it, and when specified, displays the data at an operator station.

NCP provides an interface that allows NPM to collect performance statistics for resources that require user-written line control. This interface allows the IBM special products or user-written code access to information about activity involving its resources on an NCP-NPM session. NCP notifies the IBM special products or user-written code to supply its statistics when building the COLLECT RU. NCP maintains the COLLECT RUs, but each resource for IBM special products or user-written code must build its own resource record in the RU.

User Line Control (Interrupt Level Interface)

NCP uses the XIO interface to communicate to IBM special products or user-written code NPM requests to start or stop collecting statistics on one of its resources. If specified by IBM special products or user-written code, NCP also communicates an NPM FORWARD request. Table 16 lists the command modifiers for SETMODE.

Table 16. SETMODE Modifier Descriptions

Command Modifier	Value	Description
NPM Capability	X'01'	Verifies that NPM activity is supported. Return code 0 indicates support of NPM.
NPM START	X'02'	Indicates a START request received for the resource.
NPM FORWARD	X'03'	Indicates a FORWARD request received for the resource if an NCHNG was invoked to request notification.
NPM STOP	X'04'	Indicates a STOP request received for the resource.
NPM COLLECT	X'05'	Indicates the collected data should be copied into the resource record in the COLLECT PIU.

The following sections describe the processing for these command modifiers.

Start Processing for User-Written Line Control

When NCP receives a START PIU, NCP verifies its validity, including validating each request in the START RU. For performance data collection to begin on a specific resource, the resource must be defined as eligible for NPM performance data collection. For user-written line control resources, NCP issues an NPM activity notification (SETMODE) with the Capability modifier. A return code of 0 from the user-written line control indicates that NCP supports NPM data collection for the IBM special products or user-written code resource.

When eligibility is established, start processing begins on the resource. NCP retrieves an NQE from the NQE pool and formats it for the resource. NCP sets up the element address and type byte.

IBM special products or user-written code use the NCHNG macro to update an NCP control block that the IBM special products or user-written code cannot access. The NCHNG macro includes the NQE in the list of valid control blocks and uses the keywords listed in Table 17 to change the length of the resource record in the NQE and change the notification bit.

Table 17. NCHNG Keyword Descriptions

Keyword	Invocation rules	Description
CNTBLK	Must be a register that points to the NQE; must not be register 1.	A control block to modify
COLLEN	Must be an odd-numbered register; can be a byte register; must not be register 1.	The total length of the resource record including 4 bytes that NCP sets up
FORREQ	FORREQ=SET FORREQ=RESET	SET/RESET notification of forward processing

The user-written line control resource receives control when NCP issues an NPM activity notification to that resource with a Start modifier. The user-written line control resource begins its start processing, issues the NCHNG macro to set the length of the resource record, and sets notification processing.

Forward Processing for User-Written Line Control

NCP builds a COLLECT PIU header to begin collect processing. It checks to determine which resources to collect data from: IBM special products or user-written code, NCP, or both. When NCHNG FORREQ=SET is issued on the resource's NQE control block, NCP issues an NPM activity notification with the Forward modifier to IBM special products or user-written code. The activity notification alerts the resource that forward processing has begun. When all of the resources to be collected are considered, NPM collect processing begins.

Stop Processing for User-Written Line Control

When NCP receives a STOP PIU from NPM, NCP issues an NPM activity notification to the IBM special products or user-written code with the Stop modifier. NCP returns the NQE to the NQE pool.

Collect Processing for User-Written Line Control

NCP collects data for all NCP resources and forwards it to NPM. Forward notification is issued to all resources when FORREQ=SET. After waiting two seconds, NCP collects performance data from all resources for IBM special products or user-written code. The resources for IBM special products or user-written code must copy their data into the COLLECT RU. This data is the total resource record defined by NPM excluding the first 4 bytes. These bytes contain the element address (2 bytes), the type byte (1 byte), and the length of the resource record (1 byte).

Note: The length of the resource record is the total length including the 4 bytes that NCP sets up.

NCP gives IBM special products or user-written code control for its own resource collection by issuing an NPM activity notification with the Collect modifier. When it receives control, the IBM special products or user-written code finds the buffer to begin writing into at NQEBUFP of the NQE control block. This buffer may already contain data, so you must use the DATCNT and OFFSET fields to find the next available byte. If the data does not fit into one buffer, you must span the data to the next buffer provided in the chain. NCP calculates the buffer requirement based on the length of the resource record specified in the NQE set up by the IBM special products or user-written code during start processing.

The IBM special products or user-written code resource must update the buffer count in each buffer into which it writes. NCP validates the counts, updates the transmission header count, and increments the number of resource records in this COLLECT RU. If the counts calculated by the IBM special products or user-written code and NCP do not match, NCP indicates that the collection failed by sending NPM only the 4-byte header, which NCP set up.

Using NPM Session Accounting

You can use NPM to provide accounting management for IBM special products or user-written code resources. The NPM function collects the accounting information, logs it, and when specified, displays the data at an operator station.

NCP notifies the user-written accounting notification routines when the accounting send mode changes or when NPM requests that all accounting statistics collected be sent to NPM.

NCP can give the user-written accounting notification routines the following specific types of notification:

- Send mode enabled (X'01')
- Session data requested (X'02')
- Send mode disabled (X'03')
- Report accounting thresholds (X'04')
- Change accounting thresholds (X'05').

The user-written accounting notification routine can use the following macros to support the notification processing:

NPAQSTAT	Retrieve boundary session accounting function status.
NPAQINFO	Retrieve LU-LU session information.
NPAPIU	Send NPA PIU to NPM.
NEOAXT	Accounting notification routine exit to NCP.

Use the NPAQSTAT macro to allow the user-written code to query the status of the boundary session accounting function. This macro also indicates whether the NCP accounting data send mode is enabled or disabled.

The NPAQSTAT macro returns one of the following statuses:

- Accounting collection is included and accounting data send mode is enabled.
- Accounting collection is included and accounting data send mode is disabled.
- Accounting collection is not included.

Use the NPAQINFO macro to retrieve SNA LU-LU session information for user-written line control resources. You can retrieve information about the logical unit, session partner logical unit, or procedure correlation identifier.

The NPAQINFO macro returns one of the following statuses:

- Requested information has been returned.
- Requested information has been returned; the logical unit is not being accounted for by the boundary session accounting function.
- Requested information has not been returned; NCP does not include the boundary session accounting function.
- Program correlation identification information is not available.

Use the NPAPIU macro to send an NPA PIU (accounting data, session start, session end, or session start/end request/response units) to NPM.

The NPAPIU macro returns one of the following statuses:

- PIU sent to NPM.
- PIU is not a FID1, FM data PIU.
- PIU cannot be sent; data send mode for user accounting is disabled.
- PIU sent to NPM; however, the data send mode for user accounting is disabled because of a held virtual route.
- NCP does not support the boundary session accounting function.

Use the NEOAXT macro to exit from the accounting notification routine to NCP.

Send Mode Enabled Processing

Your accounting notification routine gets control from NCP when an ENABLE, SDT, UNBIND, or ANS request is issued. NCP calls the IBM special products or user-written code when the following conditions are met:

- Boundary session accounting is enabled.
- The primary NPM for boundary session accounting supports accounting for IBM special products or user-written code.
- The virtual route for the primary NPM for boundary session accounting is not held.

If the first two conditions are met but the virtual route for the primary NPM for boundary session accounting is held, NCP calls the IBM special products or user-written code when the virtual route becomes unheld. If your routine has not already begun collecting accounting data, you can use this notification to initiate collection.

Session Data Requested Processing

NCP notifies your accounting notification routine when a SOLICIT request is received and the accounting data send mode is enabled.

Note: If the send mode is not currently enabled, this notification indicates that the send mode is being enabled.

Your accounting notification routine should build an NPA PIU using the NPM accounting control vectors (CV80 through CV8F). Use the NPAPIU macro to send the PIU to NPM. The solicit request is not totally satisfied until the NPM PIUs are accepted by the NPAPIU macro. If the PIU is not accepted because the data send mode is disabled, the IBM special products or user-written code should hold the information and send it when the data send mode for user accounting is enabled.

Send Mode Disabled Processing

Your accounting notification routine gets control from NCP when an SDT, UNBIND, or ANS request causes the accounting data send mode to be disabled. NCP calls the IBM special products or user-written code when it receives a DISABLE request or when the virtual route for the primary NPM changes to held status.

Report Accounting Thresholds Processing

NCP sends an NPM activity notification to the IBM special products or user-written code when it receives a QUERY RU from NPM or when a TAKEOVER NOTIFICATION RU needs to be sent to NPM. Your accounting notification routine must give NCP its current accounting threshold values. NCP passes a buffer to your routine to build the user accounting parameters control vectors. The threshold control vectors (CV90 through CV9F) must fit within one buffer passed from NCP.

Change Accounting Thresholds Processing

Your accounting notification routine gets control from NCP when it receives a CHANGE RU from NPM. The accounting notification routine must interpret the accounting parameter control vector (CV90 through CV9F) and update the accounting threshold values.

Formatting Customized Control Blocks

You can use the independent dump utility to format blocks for user line control code if you have identified the user blocks in a block dump table (BDT). Three macros, GRPENTRY, BLKENTRY, and GRPEND, are furnished to help in building the BDT. You need one BDT for each user-controlled line group you want to format.

Block Dump Table Requirements and Characteristics

Two types of user blocks can be described in the BDT for formatting: control blocks with a fixed length, and tables with lengths that depend on the network configuration.

The requirements for and characteristics of a BDT for user line control are as follows:

- The GCB associated with a line group must contain the BDT pointer at offset 92 (X'5C').
- The first entry in the BDT is a UACB pointer as defined by the GRPENTRY macro.
- Each user block to be formatted must be described in the BDT.

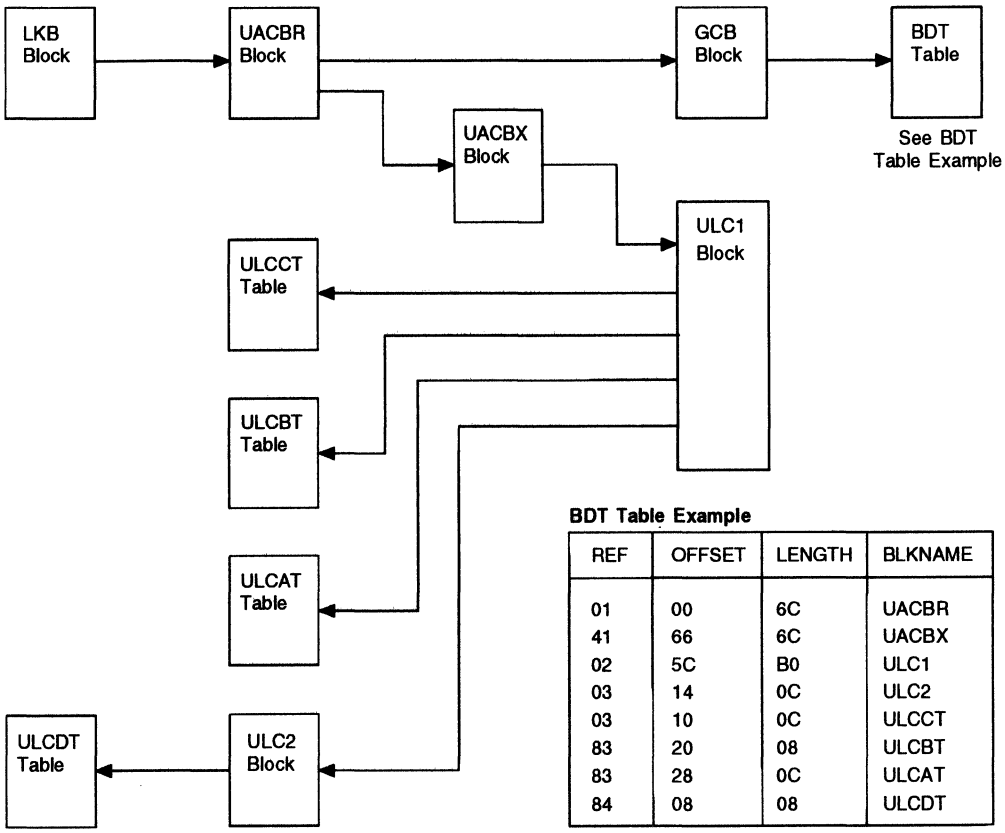
- Each user block that is described in the BDT must be pointed to by another block (its parent block).
- The BDT byte count is a 2-byte field at the BDT address minus 2.

Example of a Block Dump Table

Table 18 shows the format of a single BDT entry. Figure 13 on page 71 is an example of the BDT layout for the UACBs and their associated blocks.

Table 18. Format of the Block Dump Table

Code Number	Contents	
0(0)	x...	When bit 0 is 1, the entry describes a variable-length table. The number of table elements is stored at the table address minus 2. When bit 1 is 0, the entry describes a fixed-length control block.
	.x..	When bit 1 is 1, the address is contained in a halfword. When bit 1 is 0, the address is contained in a fullword.
	..xx xxxx	This is the byte offset within the BDT of the parent block that contains the pointer to the block being described.
1(1)	This is the byte offset within the parent block where the pointer to the current block is located. This byte is not used in the first entry. X'FF' indicates that the current block is appended to the parent block.	
2(0)	This is the length of the current control block or, for a table, the length of one entry in the table.	
3(0)	This is the 5-character name of the formatted block. It is used as the header for the current block in the formatted dump, for example, UACBR, UACBX.	



BDT Table Example

REF	OFFSET	LENGTH	BLKNAME
01	00	6C	UACBR
41	66	6C	UACBX
02	5C	B0	ULC1
03	14	0C	ULC2
03	10	0C	ULCCT
83	20	08	ULCBT
83	28	0C	ULCAT
84	08	08	ULCDT

Figure 13. Example of a Block Dump Table and Associated Control Blocks for User Line Control

IBM SDLC 3270 Bracket State Management

A bracket defines a sequence of PIUs between the host processor and an IBM SDLC 3270 Information Display System, and enables data to be transmitted without interruption. A beginning-of-bracket PIU and an end-of-bracket PIU start and end the sequence. Flags in the request/response header identify a PIU as a beginning-of-bracket PIU or an end-of-bracket PIU. When a bracket consists of only one PIU, both the beginning-of-bracket and the end-of-bracket flags in the PIU are set.

The BIND command (refer to the section on network commands in *NCP and EP Reference*) specifies whether or not the session is to use brackets. If so, brackets must be used throughout the session; no data may be transmitted except within a bracket. Since the host and the IBM 3270 system may not concurrently send brackets, the following protocol defines the process by which one unit may gain control of the line.

The IBM 3270 system always gains control if it and the host concurrently contend for the line. It also may begin a bracket without permission from the host. To begin a bracket, the IBM 3270 system simply transmits a PIU. NCP sets the beginning-of-bracket flag in the PIU and relays it to the host. The host, however, must obtain permission from the IBM 3270 system to begin a bracket. It seeks permission by attempting to initiate a bracket by sending a beginning-of-bracket

PIU. The IBM 3270 system grants or refuses permission depending on whether or not it has data to transmit.

Bracket States

A bracket state manager (BSM) within NCP supervises the process of establishing the in-bracket state. Bracket protocol may require several changes of state in the BSM before the in-bracket state is established or discontinued. These states are maintained in the LUBASSET field of the logical unit control block. An understanding of the following bracket states is crucial to understanding the flow of data in bracket mode.

Between Brackets: Neither unit is currently using or bidding for use of the line.

Between Brackets/Bid Pending (BETB/BIDP): If, while in the between-brackets state, NCP receives a beginning-of-bracket PIU from the host, it retains the PIU and seeks permission from the IBM 3270 system to begin the bracket. It seeks permission by issuing a pseudo-bid PIU. When NCP sends this pseudo bid, the BSM state changes from between brackets to BETB/BIDP.

Between Brackets/Beginning-of-Bracket PIU Pending (BETB/BBP): If the IBM 3270 system returns a positive response to the pseudo bid, NCP relays a beginning-of-bracket PIU. The IBM 3270 system still has the opportunity to poll its devices and, if it acquires data to be transmitted, to refuse the bracket. When the application sends the beginning-of-bracket PIU, it may request a definite response or an exception response.
If the application requests a definite response, the IBM 3270 system returns a response indicating whether or not it will accept the bracket. NCP indicates that a response is pending by changing the BSM state to BETB/BBP. If the application requests an exception response, the 3270 system will return only if it will not accept the bracket.

In-Bracket (INB): A unit has obtained possession of the line and is transmitting a bracket.

In-Bracket/Bid Pending (INB/BIDP): The 3270 system should not send data while the BSM is in BETB/BIDP state. However, the 3270 system occasionally sends data after NCP sends a pseudo bid before its arrival in the 3270 system. When NCP receives the data PIU, it sets the beginning-of-bracket flag, relays the PIU to the host, and changes its state to INB/BIDP.

Figure 14 on page 73 through Figure 18 on page 76 illustrate the possible flows of traffic through, and changes of state in, the BSM.

After the bracket state is established, data flows within the bracket until the initiating unit tries to end the bracket. The 3270 system cannot issue an end-of-bracket PIU, but indicates the end of a bracket in the data it transmits. The host, whether terminating its own bracket or receiving an end-of-bracket indicator from the 3270 system, issues an end-of-bracket request PIU to NCP, which in turn immediately changes the BSM state to between brackets. This immediate state change is called *unconditional end-of-bracket* mode.

Exception Conditions

The BSM identifies two types of exception conditions associated with bracket management:

Send Bracket State Error (X'2003') indicates a protocol error when the host BSM and the 3270 BSM are out of synchronization. The BSM discards the request and notifies the function manager.

Send Bracket Contention (X'0813') indicates the refusal of the 3270 system to allow the host to send a bracket. The BSM discards the request.

The following figures show exception conditions as they may occur during bracket state management.

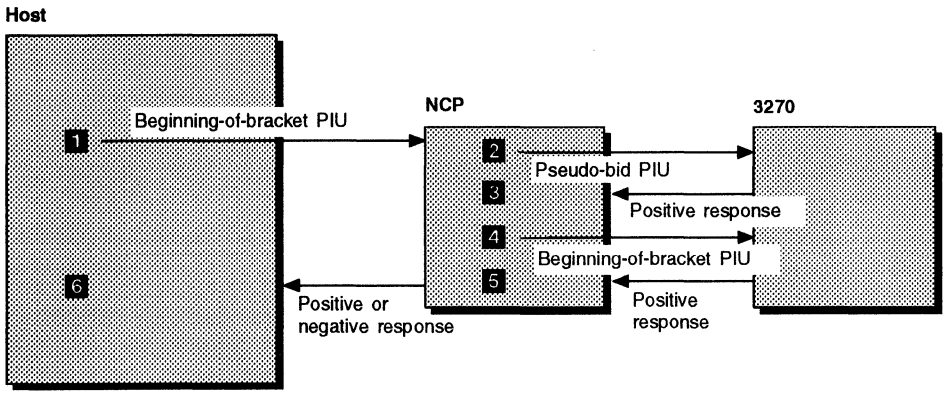


Figure 14. Host's Initiation of a Bracket when the 3270 System Has No Data to Transmit

The following comments describe the data flow sequence illustrated in Figure 14.

1. The BSM is in the between-brackets state. The host seeks permission to send a bracket by sending the beginning-of-bracket PIU.
2. NCP builds a pseudo bid (X'F8'). This bid is NCP's request to the 3270 system to begin the bracket. The BSM state changes to between-brackets and bid pending.
3. If the 3270 system does not have an attention pending, it stops polling and returns a positive response to the pseudo bid.
4. NCP sends the beginning-of-bracket PIU to the 3270 system. If a definite response is requested, the BSM state changes to between-brackets and bid pending. If only an exception response is requested, the state changes to in-bracket.
5. If there is still no attention pending, the 3270 system returns a positive response to the beginning-of-bracket PIU (if requested) and resumes polling. Otherwise, it returns a negative response.
6. NCP changes the BSM state to in-bracket (if the response is positive) or between-brackets (if the response is negative) and returns the response to the host. This establishes the bracket state.

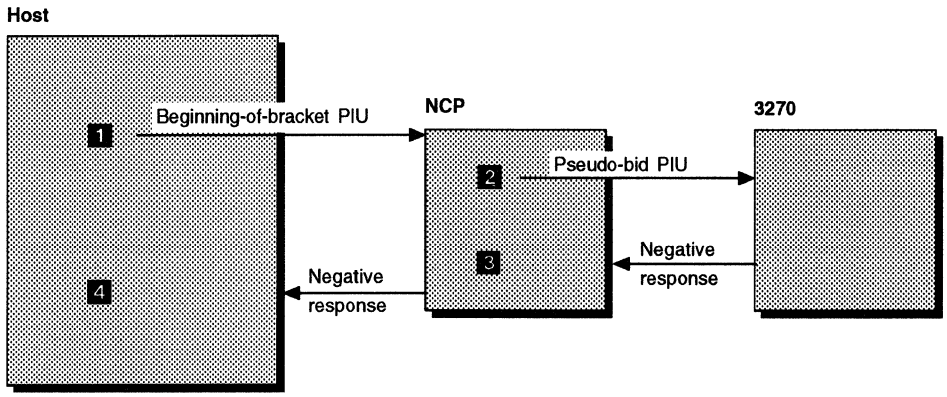


Figure 15. Host's Attempt to Initiate a Bracket when the 3270 System Has Data to Transmit (3270 Attention Pending)

The following comments describe the data flow sequence illustrated in Figure 15.

1. The BSM is in the between-brackets state. The host seeks permission to send a bracket by sending the beginning-of-bracket PIU.
2. NCP sends a pseudo bid and the BSM state changes to between-brackets and bid pending.
3. The 3270 system has an attention pending and returns a negative response to the pseudo bid.
4. The BSM state reverts to between-brackets and NCP returns the negative response to the host.

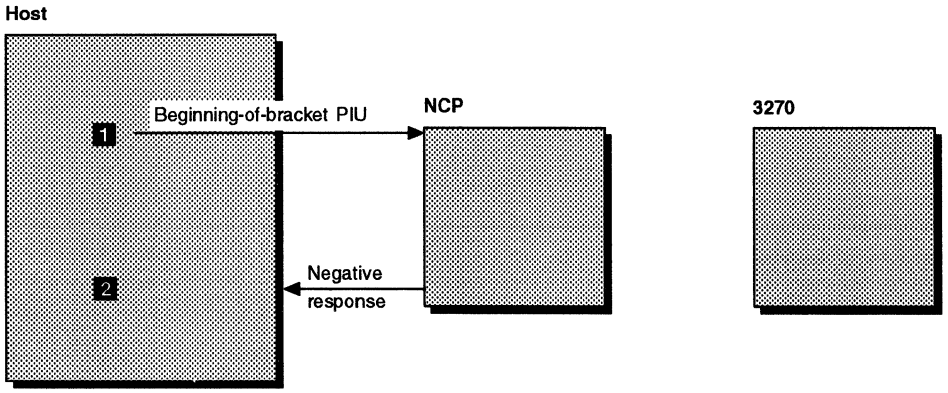


Figure 16. Host's Attempt to Initiate a Bracket when the 3270 System Has Data to Transmit (Bracket State Active)

The following comments describe the data flow sequence illustrated in Figure 16.

1. The BSM state is in-bracket and the 3270 system is transmitting a bracket. The host attempts to initiate a bracket.
2. The BSM, already in the in-bracket state, returns a negative response to the beginning-of-bracket PIU.

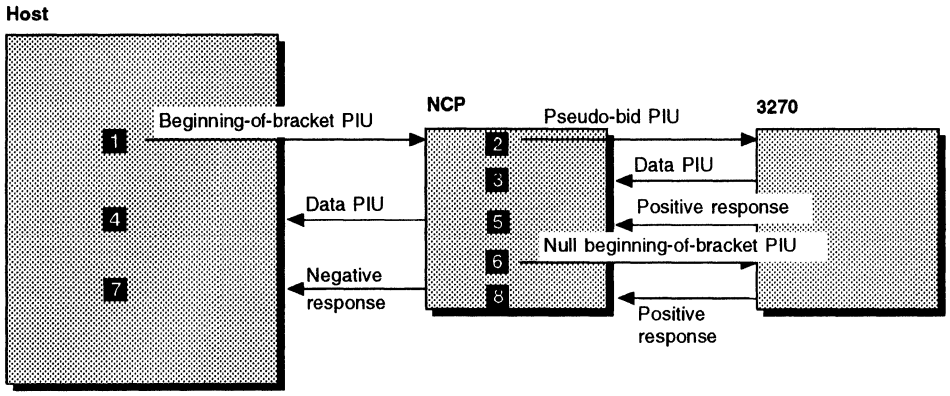


Figure 17. Host's Attempt to Initiate a Bracket when the 3270 System Initiates a Bracket before Receiving the Bid. The 3270 returns a positive bracket response.

The following comments describe the data flow sequence illustrated in Figure 17.

1. The BSM is in the between-brackets state. The host seeks permission to send a bracket by sending the beginning-of-bracket PIU.
2. NCP sends a pseudo bid and the BSM state changes to between-brackets and bid pending.
3. The 3270 sends a data PIU before receiving the pseudo bid.
4. The BSM state changes to in-bracket and bid pending and NCP relays the PIU to the host.
5. NCP receives a positive response to the original pseudo bid. At this point, the 3270 system stops polling its devices. NCP sends a null beginning-of-bracket PIU to resume polling.
6. NCP then returns a negative response to the original beginning-of-bracket PIU to the host.
7. The 3270 system resumes polling and returns a response to the null beginning-of-bracket PIU.
8. NCP receives a response to the null PIU. Since the purpose of the null PIU is to resume polling, it does not matter whether the response is positive or negative.

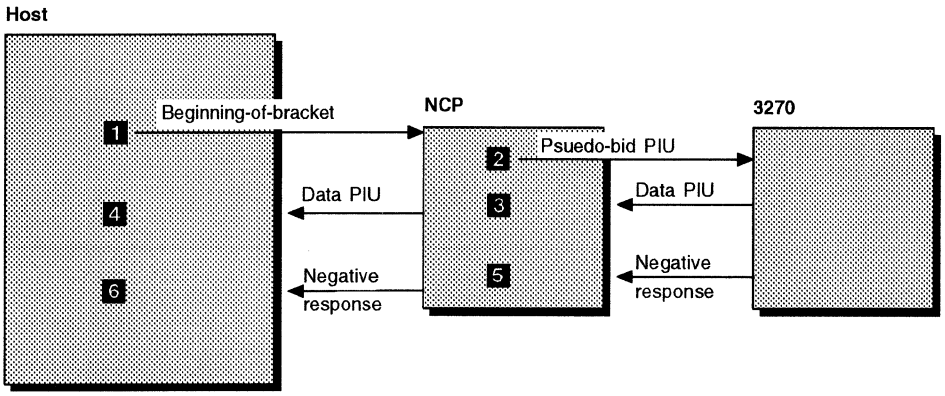


Figure 18. Host's Attempt to Initiate a Bracket when the 3270 System Initiates a Bracket before Receiving the Bid. The 3270 returns a negative bracket response.

The following comments describe the data flow sequence illustrated in Figure 18.

1. The BSM is in the between-brackets state. The host seeks permission to send a bracket by sending the beginning-of-bracket PIU.
2. NCP sends a pseudo bid and the BSM state changes to between-brackets and bid pending.
3. The 3270 system sends a data PIU before receiving the pseudo bid.
4. The BSM state changes to in-bracket and bid pending and NCP relays the PIU to the host.
5. NCP receives a negative response to the pseudo bid. The 3270 system does not stop polling, so NCP does not need to send a null beginning-of-bracket PIU.
6. NCP returns a negative response to the pseudo bid to the host.

Chapter 3. Customizing Programmed Resources

This chapter describes how to create and use programmed resources. The information in this chapter is specific to programmed resources only.

In NCP, programmed resources are physical units and logical units that reside in the controller but can be logically external to NCP. These resources can provide special-purpose functions, such as collecting traffic data, or they can provide some SSCP functions normally provided by the access method. Programmed resources can be type 1 or type 2 physical units, but cannot be type 4 physical units. Programmed resources that are logically external to NCP are referred to here as *external programmed resources* to distinguish them from *programmed network addressable units.*

Programmed network addressable units are programmed resources that provide SSCP functions and operate as part of NCP. To a terminal or to NCP, the programmed network addressable unit is an SSCP but is never recognized by the access method since it is in segment 6 of the resource resolution table. The programmed network addressable unit maintains ownership of the real resources that are being controlled by the host through the programmed network addressable unit. Programmed network addressable units must provide a LINE-PU-LU definition statement structure.

Because programmed resources must duplicate much of NCP's function and are responsible for all level 5 functions, you must be very familiar with NCP flow and control, SNA architecture, and the protocols used by any devices or line controls you may be supporting with your programmed resources.

All PIUs that are destined for a programmed resource are passed to your program; NCP takes no action with these PIUs. However, your programmed network addressable unit should not try to begin or end cross-domain sessions with resources in other domains. Rather, the host SSCP that owns the programmed resources should be allowed to control these cross-domain sessions for the programmed resources. To exercise the control, your program must first issue an Activate Physical command to NCP to identify itself and become an owner of NCP. Your program can then issue commands that enable it to own and control resources in NCP. When your program operates in this mode it must be prepared to receive network commands such as the Lost Subarea command.

Because programmed network addressable units are a part of NCP, they are not subject to automatic network shutdown (ANS). However, programmed network addressable units may receive a subsequent Lost Subarea command when a host SSCP is restarted or is shut down. If an external SSCP that owns a programmed resource is lost to NCP, the resource is shut down by ANS, the network addressable unit acting as the SSCP is notified, and you can then reset your tables.

Writing Programmed Resources

All programmed resources run in level 5 of NCP and are located logically at the junction between subarea path control and the connection point manager (CPM) in NCP's peripheral function. CPM is part of transmission control. The resource has all CPM responsibilities for sessions involving it and must also provide all level 5 functions normally provided by NCP. Your level 5 programs have a storage protection key of 1.

Programmed resources must also provide CPM function for sessions they are involved in. You can use NCP customization macros to define much of this function. "Using NCP Macros in Programmed Resources" on page 92 describes the functions provided.

CPM is the traffic controller in the peripheral function, coordinating the flow of data traffic within a session. It performs the following functions:

- Controls pacing
- Ensures session integrity
- Manages segmented path information units (PIUs)
- Manages sequence numbers
- Controls data flow.

For more information on CPMs, see *NCP and EP Reference* and *Systems Network Architecture Format and Protocol Reference Manual.*

The programmed resource support in NCP has the following characteristics:

- Trace-request PIUs are passed to the user of level 5 programmed resources.
- Trace-request PIUs for user-written line control are handled by level 5 code in NCP or by the user's code in level 5 for programmed resources, whichever is applicable.
- The current FVTABLE search argument and the next-previous search argument are stored in the control block for the affected resource.
- Control blocks for programmed resources are shown in formatted dumps of controller storage.
- Customization macros for use with programmed resources have diagnostic MNOTE messages and can use keywords.
- Customization macros for use with programmed resources can pass return codes and error bits to the invoking routine.
- All NCP panel functions through MOSS are applicable except a panel-initiated line test for lines being driven by user-written line control.

Programmed resources have the following primary responsibilities:

- To accept PIUs from other logical units or SSCPs and from level 2 or level 3
- To send those PIUs to the proper NCP element
- To maintain control of the sessions involving the programmed resources under your control.

Use the destination address field (DAF) in the PIU to find the control block that defines the destination resource. If the control block is a programmed resource

logical-unit block (NLB), the origin address field (OAF) is compared with the session partner's network address in each programmed resource logical unit block extension (NLX) to determine if a session is already in progress. If so, the PIU is queued on the appropriate NLB extension.

The NCP-physical-unit-services routine queues PIUs requesting service for that resource. Your routines must be prepared to accept these requests from the physical unit services routine for programmed resources. When handling such requests, you can use the NEOENQ macro to send any request or response PIU that requires sequence-number control and requires that the OAF contain the address of NCP physical services. You can also use the NEOENQ macro where there is an error requiring that an error indication be sent to the host.

Use *Systems Network Architecture Format and Protocol Reference Manual* to determine which PIUs are likely to be encountered by each type of resource. The following paragraphs list resources and typical PIUs.

Programmed Links: Requests such as Activate Connect In or Activate Link for physical unit services may be received on the queue for a programmed link.

Programmed Physical Units: There is a QCB for the SSCP-PU session for each programmed physical unit. All PIUs with a DAF containing the address of a programmed physical unit are queued on its QCB. The NCP dispatcher dispatches the user task that is currently associated with the QCB. The PIU may be a response, or it may be a request such as Activate Physical, Deactivate Physical, or Start Data Traffic. Requests such as RNAA or CONTACT for physical unit services relating to the resource may also be received on the queue for a programmed physical unit.

Programmed Logical Units: Each programmed logical unit has a QCB associated with it. PIUs are queued to such a QCB if they have a DAF that contains the address of a programmed logical unit and a OAF that does not match the partner network address field of any NLB extension on the NLB. Generally, PIUs on a programmed logical unit's QCB have an OAF containing the address of an SSCP. The PIU may be a response, or it may be a request such as Activate Logical, Control Initiate, or Bind Session. Depending on the SSCP, requests such as Set Control Vector for physical unit services may be received on a queue for a programmed logical unit.

Programmed Sessions: Each logical unit can have up to eight QCBs for parallel LU-LU sessions with that logical unit. All PIUs that have a DAF containing the address of this logical unit, and whose OAF matches the partner-network-address field in an NLX associated with the NLB for this logical unit, are queued to the NLX that represents this unique OAF and DAF pair. The PIU may be a response or a request such as Bind Session, Clear, or Unbind Session.

Programmed Network Addressable Unit (SSCP): If the logical unit is acting as an SSCP, the logical unit can also receive requests such as Bind Failure, Contacted, Inoperative, or Network Services Lost Subarea. The logical unit may also receive RECMS records, which the logical unit should send to the host that owns the programmed resource. *NCP and EP Reference Summary and Data Areas*, Volume 1, documents the changes to the RECMS records for this release.

Using the Function Vector Table

Each session is represented by a control block that begins with a QCB. PIUs are queued to the appropriate QCB as they arrive in NCP. As queues become activated (PIUs are placed on the queue), their associated tasks are scheduled and started by the task dispatcher. Input queues may also be activated by issuing a TRIGGER macro. See "Manipulating Task Status" on page 11 for more information about the TRIGGER macro.

You are responsible for providing these level 5 tasks for programmed resources. Use the FVTABLE macro to identify these tasks to NCP; use the NCHNG macro to change the task to be executed; and use the NPARMS macro to display the current task and the previous task. Refer to *NCP and SSP Customization Reference* for more information. Your level 5 routines must supply the following NCP functions:

- Interpretation of PIUs
- Control of polling and addressing if you are using the XIO level of control
- Peripheral network node processing
- Physical services
- Data handling.

The code associated with a task consists of a task sequencer and a number of task subroutines. The sequencer receives control when the NCP supervisor dispatches the task, and the task sequencer returns control to the NCP supervisor when the task has completed its processing. The task sequencer controls execution of the task by dequeuing the PIU found on the task's QCB, verifying the PIU, and initializing processing. The task sequencer then calls the appropriate subroutines in the correct order to process the PIU. For more information, see the chapters that address task management in *NCP and EP Reference*.

Validating a PIU

SNA validation includes validation not only of the format of the PIU but also of the sequence of PIUs, which is dependent on the activation state of other resources in the SNA resource hierarchy. The tasks controlling a programmed resource and its sequencer should be subdivided into several tasks to simplify the validation process. Each such task might contain the validation and other functions that are carried out when PIUs or block control units (BCUs) are received by the resource in various states of activation or deactivation. The following paragraphs describe such states.

Inactive: A higher-level resource in the hierarchy has not been activated. All requests are rejected.

Inoperative: A permanent error condition exists. The function cannot be performed, and all requests are rejected.

Awaiting Activation: All resources higher in the hierarchy are active. All requests other than Activate Link, Activate Physical, and Activate Logical are rejected. ACTLINK, ACTPU, and ACTLU functions are carried out as they are appropriate. The state of the resource on which these functions act is changed to active, then the resources lower in the hierarchy are placed in the awaiting-activation state if they were inactive. The state of inoperative resources that are lower in the hierarchy is not changed by these functions.

Active: The resource has been activated, and a session exists between an SSCP and the resource. The session control routine (the task sequencer) validates all requests. A Deactivate Link, Deactivate Physical, or Deactivate Logical request returns this resource to the awaiting-activation state and then returns all lower resources in the hierarchy that were not inoperative to the inactive state.

Data Flow Active: A logical unit is moved from the active state following the completion of bind processing so that the logical unit can participate in active LU-LU sessions. The logical unit can accept function management data requests. If an Unbind Session request state is used, logical units in the active state will reject function management data requests. An Unbind Session request returns the resource to the active state.

Your code can receive control from NCP in three ways:

- For PIUs whose DAF contains the address of a programmed resource, the buffer that contains the PIU is placed on a resource input queue. The user task associated with that queue is dispatched as a level 5 NCP task. A QCB is defined by NCP generation for each of the following sessions: one QCB for each SSCP-PU session, one for each SSCP-LU session, and one for each possible parallel LU-LU session. A QCB is also defined for each programmed link. FID0 and FID1 PIUs are queued to these input QCBs by NCP subarea path control or by the XPORT service routine.
- Your code receives control for function management data PIUs whose destination is NCP. This implies that they are requests for physical unit services and apply to a programmed resource. The PIU is placed on the resource input queue of the affected resource. Then the user task associated with that task is dispatched as a level 5 NCP task.
- NCP may force the notify task to be dispatched (triggered) when a system problem or a network problem with a programmed resource has been detected. The notify task can also receive control if it is triggered by another piece of user code.

Coding Level 5 Routines

You can use any register in a task routine. On entry to the task, register 2 contains the address of the NCP control block that defines the programmed resource requiring service. The first portion of the control block is the QCB that contains the PIU to be handled, if there is one. The address of the control block is also required when you wish to change or to get information from storage areas that are protected by NCP. Register 6 contains the address of the first level 5 save area. User-written code cannot use LA, BAL, or BLG instructions that reference labels that are above the 4MB boundary.

NCP and SSP Customization Reference describes the NCP supervisor macros your routines can use. Your task must return control to NCP by using the SYSXIT macro.

Writing a Notify Task

You must specify a notify task in the FVTABLE for each programmed resource. Your notify task is dispatched via the QCB whenever there is an unfavorable condition such as VRINOP or ANS in the network that directly affects the use of the resource. When this happens, the notify task is also dispatched to the system generation options with the status of the notify-immediate bit. Use the NPARMS macro to read the notify-task information byte in the control block that defines the programmed resource. The address of that control block is in register 2. Refer to *NCP and EP Reference Summary and Data Areas*, Volume 1, for bit definitions.

You are responsible for clearing the information byte by using the NCHNG macro when your processing is complete. For the notify task to receive control properly, you must issue the RSLVSNP macro and pass the returned SSCP-to-NCP session control block (SNP) mask to NCP using the NCHNG macro whenever a VLB, NLB, or NPB is activated. You can also use the NPARMS macro to get the current SNP mask. The SNP mask identifies the SSCP that owns the resource.

If ANS occurs, the PIUs are removed from the VLB, NPB, and NLB that originated from the lost SSCP. Also, the notify task is driven for the VLB, NPB, NLB, and NLX. One chain at a time is processed starting with the NPB and ending with the NLX. This process goes up and down the associated VLB, NPB, NLB, and NLX chains until all the control blocks have been serviced.

Managing Switched Resources

No facilities are supplied to change the structure of the programmed resource's control blocks, so it is not possible for you to change the structure during operations. For switched SDLC lines, you might receive an Assign Network Address (ANA) request from the SSCP, but you do not have the means of handling that request. Therefore, if programmed resources are to appear to be switched, the SSCP must be specified to use only a Request Network Address Assignment (RNAA) request when requesting programmed resources.

Programmed Resource Control Blocks

Programmed resources are attached to the network through several control blocks supplied specifically for that reason. These control blocks replace and are similar to the LKB, CUB, and LUB control blocks that NCP uses to control its resources.

The following control blocks are for programmed resources:

- The virtual link block (VLB)
- The physical unit block (NPB)
- The logical unit block (NLB)
- The logical unit block extension (NLX).

The VLB is generated from keywords specified on the LINE definition statement and controls a programmed resource that is the equivalent of a line. The NPB is generated from keywords specified on the PU definition statement and controls a programmed resource that is the equivalent of a physical unit. The NLB is generated from keywords specified on the LU definition statement and controls a programmed resource that is equivalent to a logical unit. The NLX is also generated from keywords specified on the LU definition statement. There is one NLX (up to a

maximum of eight) for each parallel session that the logical unit can participate in. For switched lines, the NLB and NLX are generated from keywords on the PU definition statement.

The RVT entry for a programmed resource is identified by bit 1 in the RVTTYPE2 field being set to 1 and by the standard SDLC resource bits. The following topics show the control block relationships for programmed resources.

Start-Stop and BSC Line Control

Line control for start-stop and BSC lines involves code in levels 2, 3, and 5 of NCP. The level 5 code analyzes and directs the line control, which is primarily executed in levels 2 and 3.

Figure 19 on page 84 shows the relationships among control blocks used for line control without user line control.

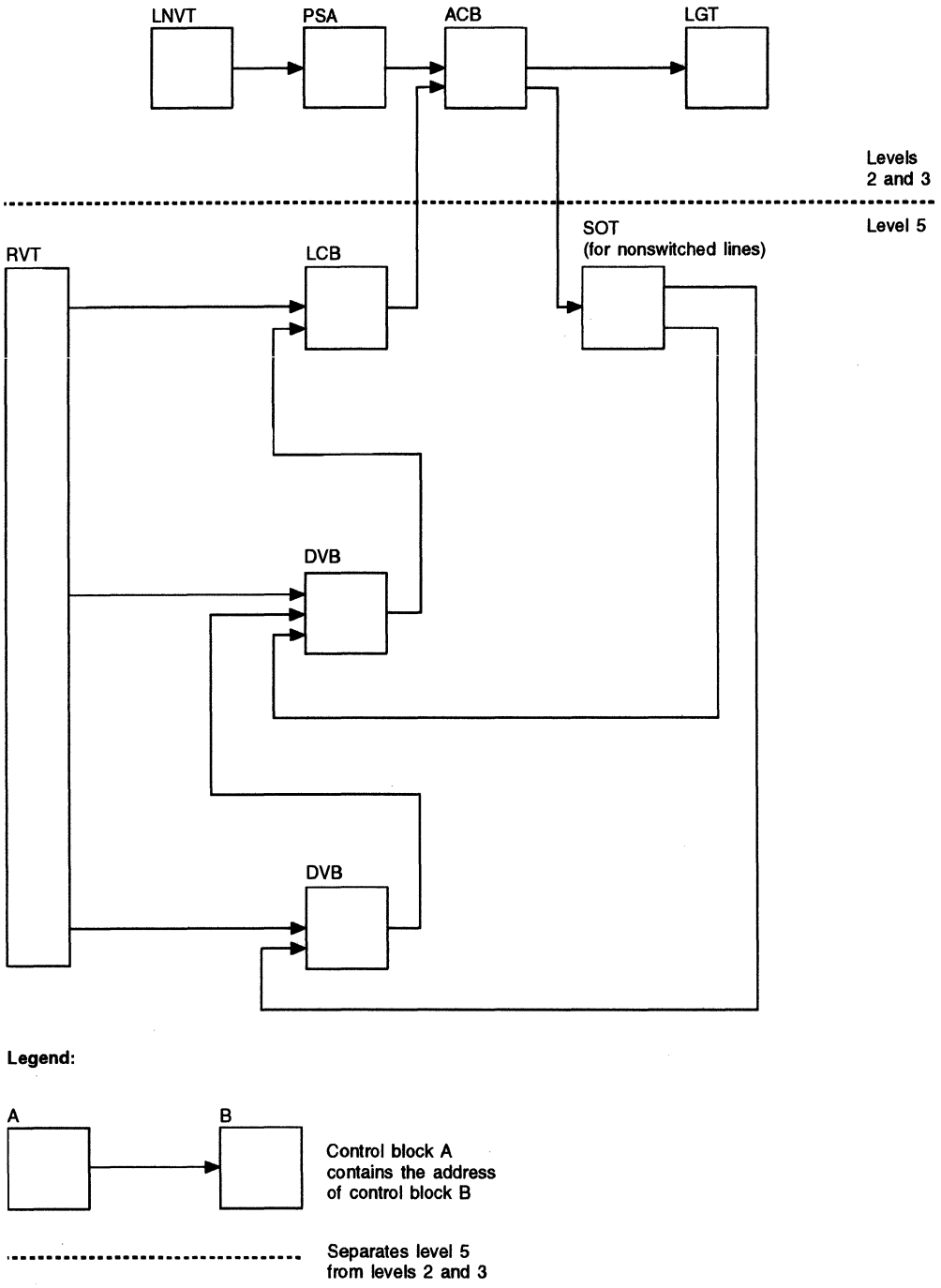


Figure 19. Control Blocks for Start-Stop and BSC Line Control without User Code

The RVT, LCB, DVB, and SOT blocks are used in level 5 processing. The LNVT, ACB, and LGT control blocks are used in level 2 and level 3 processing. A complete description of the parameter status area (PSA) control block can be found in *NCP and EP Reference Summary and Data Areas*, Volume 1.

Figure 20 shows how these relationships are altered when the user handles line control in level 2 and level 3.

Note: The level 5 control block structure is unchanged. The group control block (GCB) contains the addresses of the user routines.

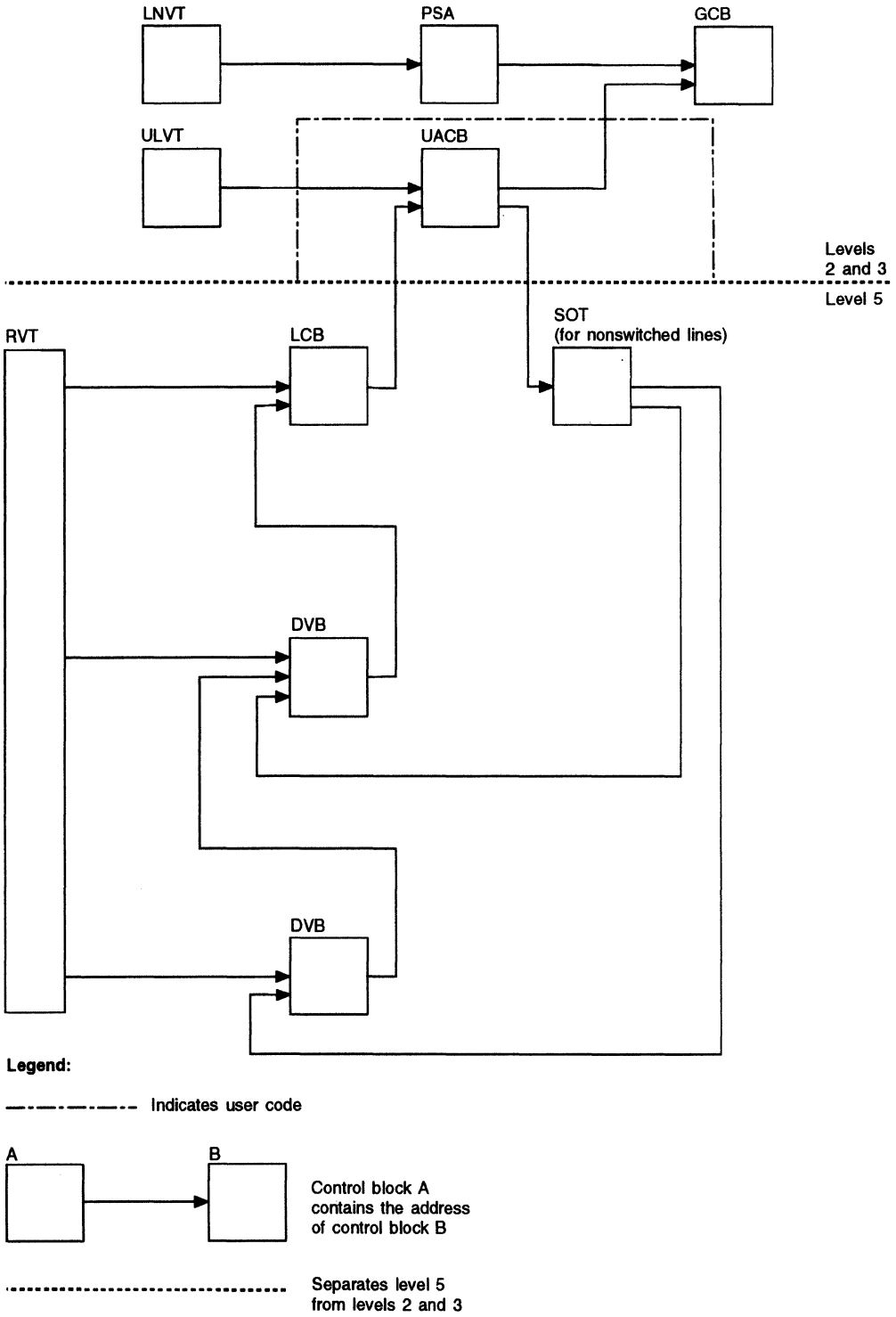


Figure 20. Control Blocks for Start-Stop and BSC Line Control, User Levels 2 and 3

SDLC Line Control

Line control for SDLC lines involves code in levels 2, 3, and 5 of NCP. Figure 21 shows the relationships between control blocks when user line control is not used.

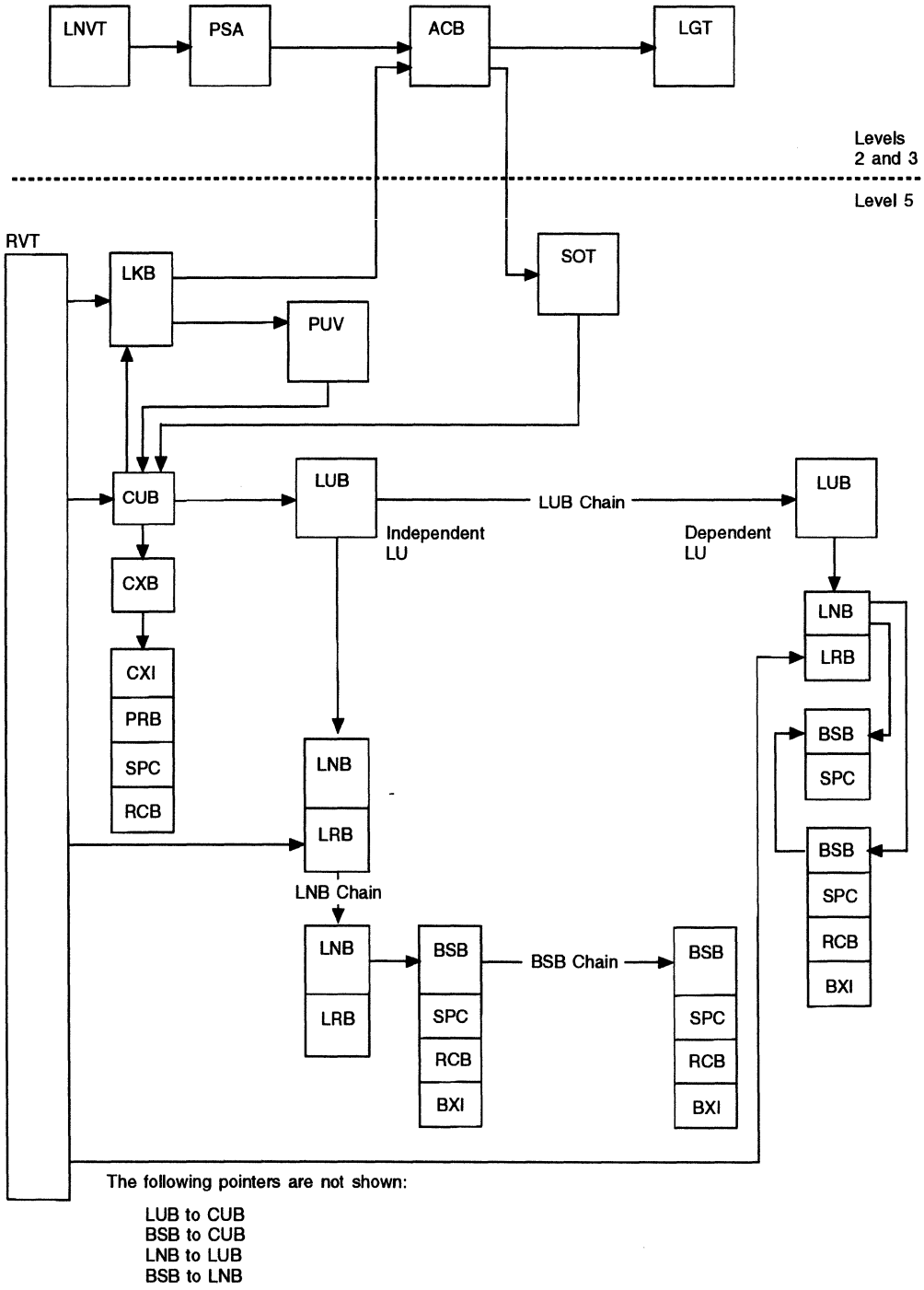


Figure 21. Control Blocks for SDLC Line Control without User Code. Note that the BSB and BXI are not contiguous when created dynamically.

The level 5 control blocks are RVT, LKB, CUB, LUB, SOT, PUV, LNB, LRB, and BSB. LUBs continue to represent logical units, but a chain of LNBs is attached to the LUB for each logical unit to represent the network addresses defined for that logical unit. Each LNB then has an associated chain of peripheral session blocks (BSBs) to represent the sessions for that logical unit.

The LNVT, ACB, and LGT control blocks are used in level 2 and level 3 processing.

Figure 22 on page 88 shows the corresponding control block relationships when you handle line control in level 2 and level 3. Again, the level 5 control block structure is unaffected.

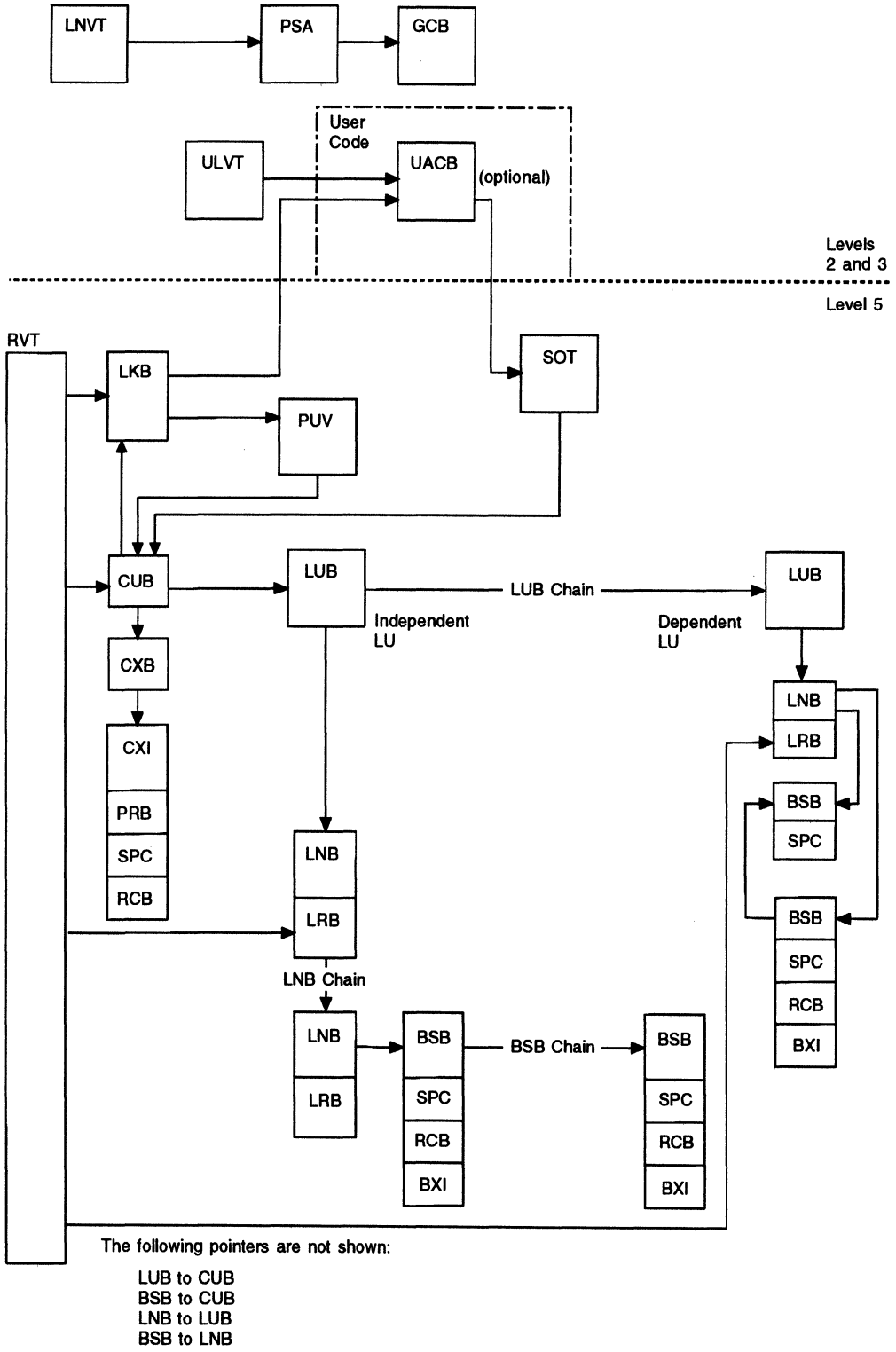


Figure 22. Control Blocks for SDLC Line Control, User Levels 2 and 3. Note that the BSB and BXI are not contiguous when created dynamically.

Figure 23 shows the control block structure when you handle line control in levels 2, 3, and 5. The user control blocks can be used in any way desired to form the interface between the programmed resources and the terminals.

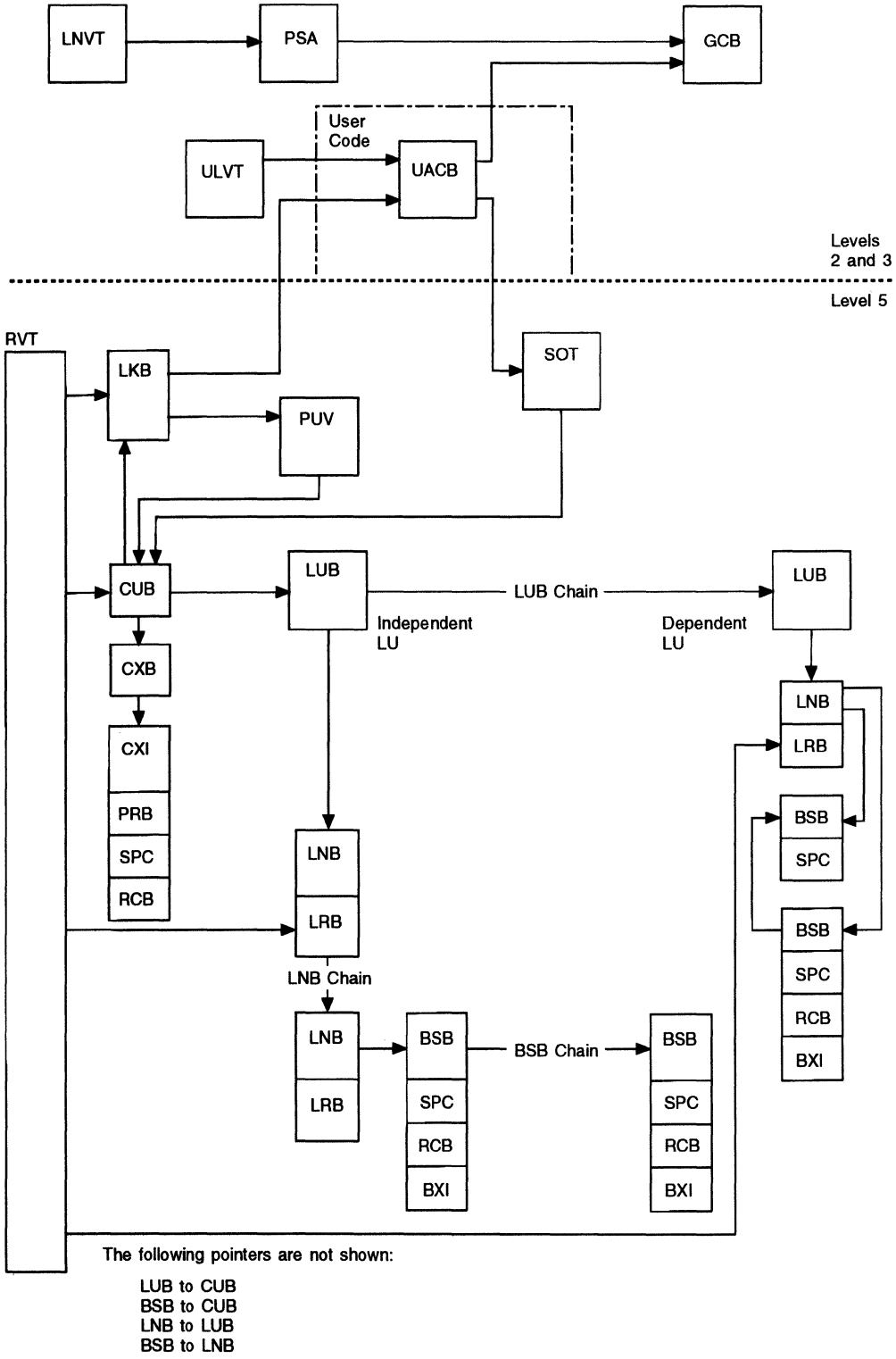


Figure 23. Control Blocks for SDLC Line Control, User Levels 2, 3, and 5. Note that the BSB and BXI are not contiguous when created dynamically.

SDLC with Programmed Resources

Figure 24 shows the control block relationships for SDLC line control. This structure is the same for switched and nonswitched lines. Figure 25 on page 91 shows the control block structure for handling programmed resources in level 5.

Note: The level 2 and level 3 control blocks (the LNVT and the control blocks connected to it) are not used with programmed resources.

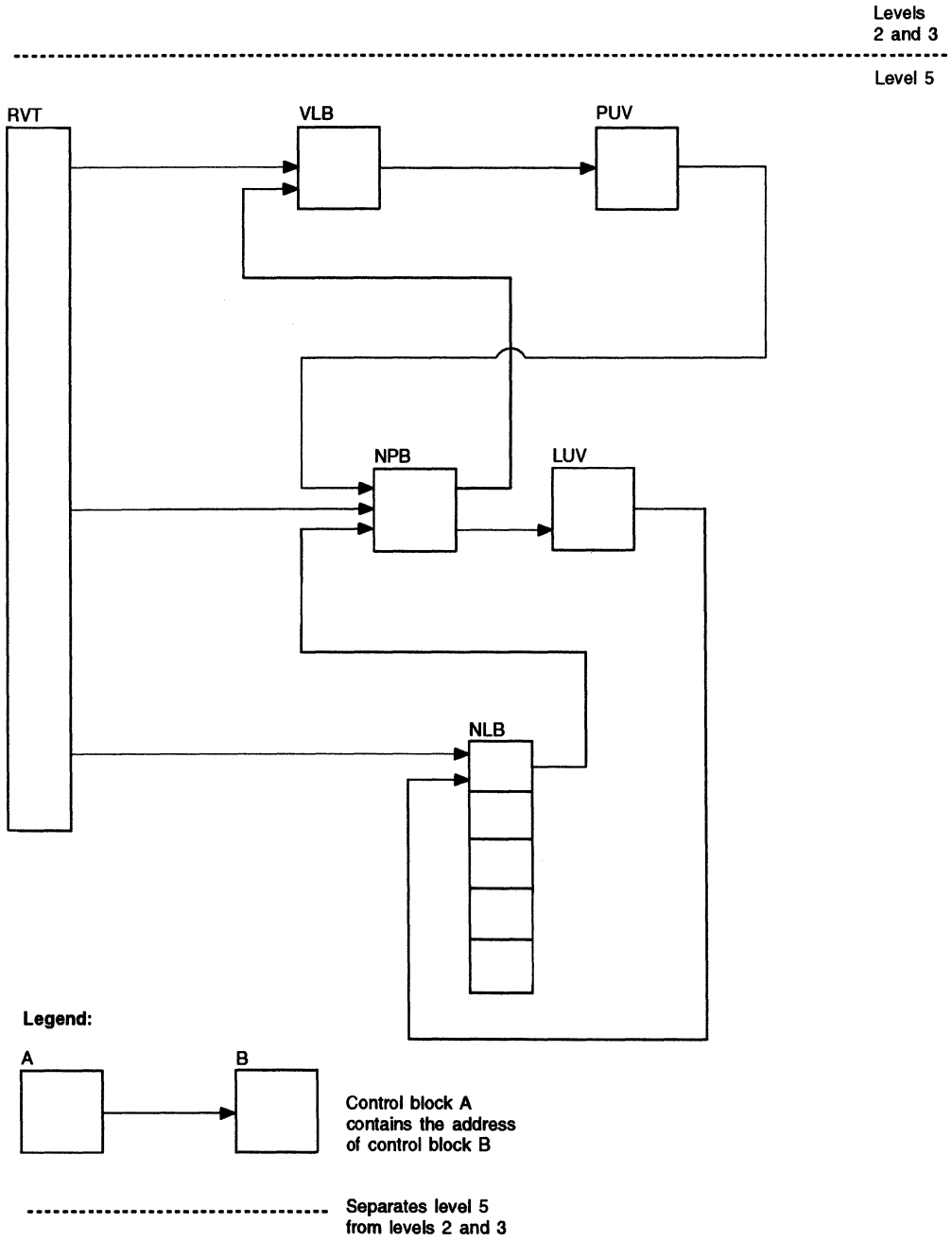


Figure 24. Control Blocks for SDLC Line Control, Programmed Resources

SDLC with Programmed Network Addressable Units

Figure 25 shows the control block structure for handling programmed NAUs. There is one NLB for each programmed network addressable unit.

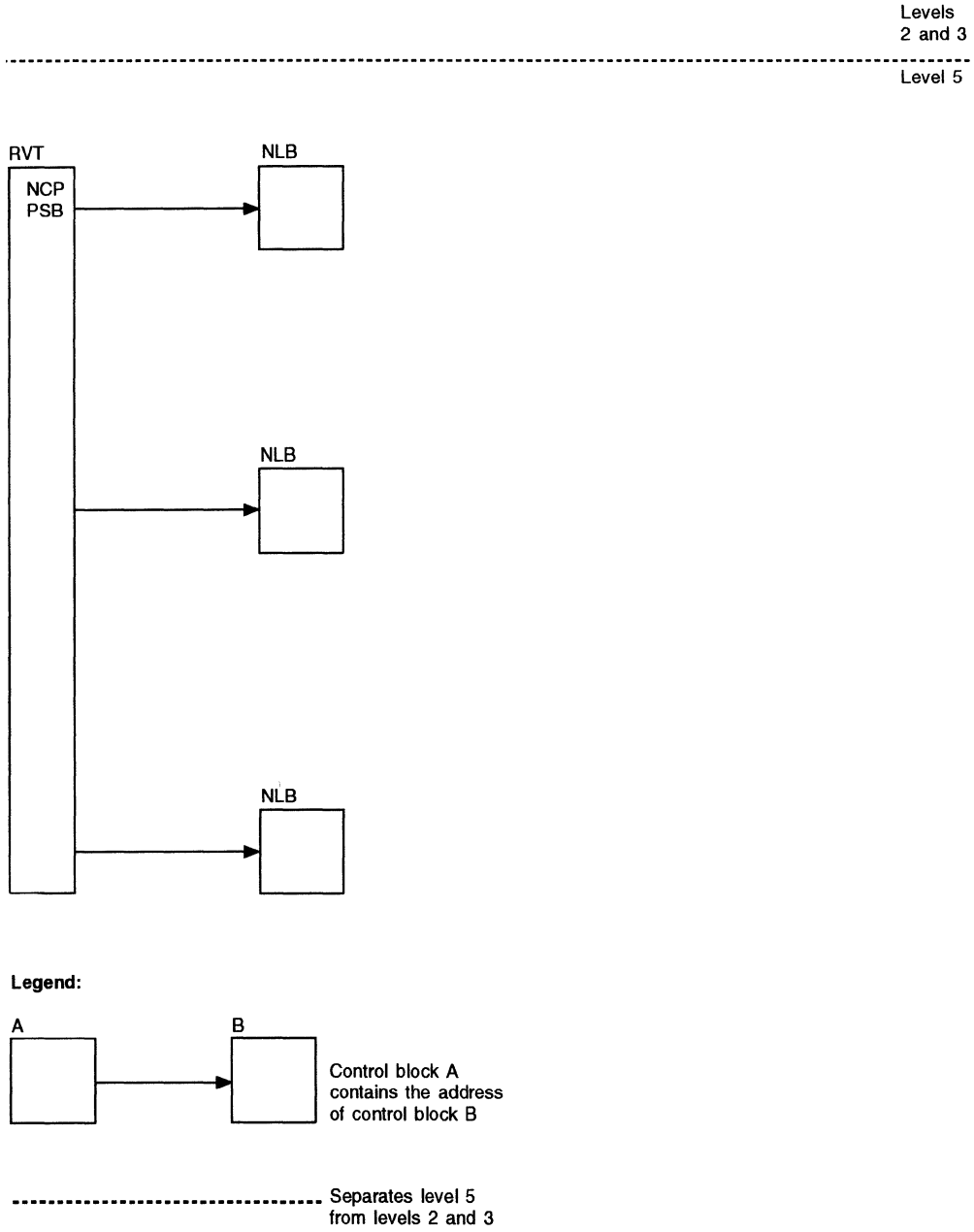


Figure 25. Control Blocks for Programmed Network Addressable Units

Vector Tables for Programmed Resources

Programmed resources are assigned network addresses within the NCP subarea during NCP generation. An address is assigned for each resource (link, physical unit, logical unit, or NAU). The control blocks for programmed resources are linked through vector tables that give their network hierarchy. Figure 26 shows this linkage.

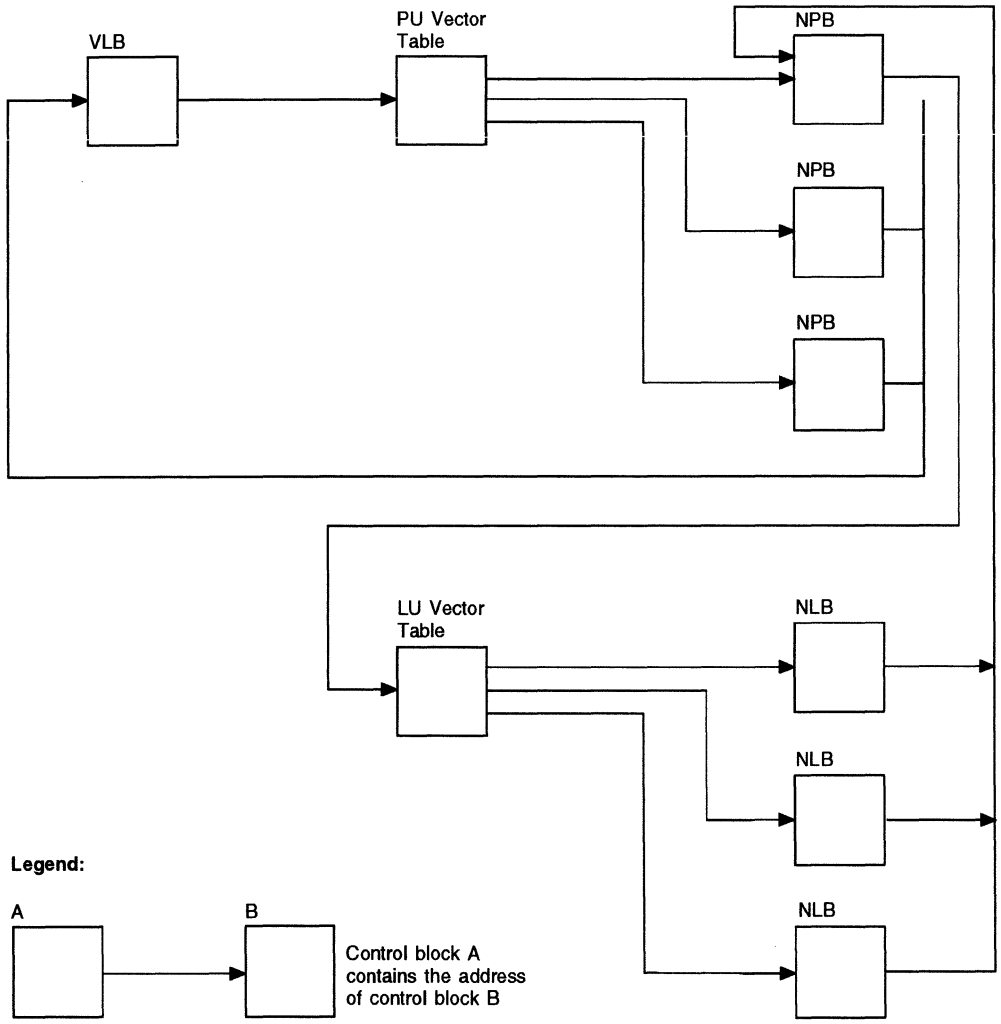


Figure 26. Vector Tables for Programmed Resources

Each NPB has a logical unit vector table and an NLB structure as shown for the first NPB.

Using NCP Macros in Programmed Resources

This section describes the functions of macros that can be used in level 5 user routines. For more detail on these macros and for a description of their keywords, refer to *NCP and SSP Customization Reference.*

Defining the Tasks for Each Programmed Resource (FVTABLE Macro)

For each programmed resource, a QCB is built during NCP generation. The tasks that may be associated with the resource are listed in a table called a function vector table (FVT). The address of the FVT is placed in the non-QCB portion of the programmed resource's control block. The current task entry point address is stored in the QCB. The FVTABLE macro generates the FVT. Each task is identified in the FVT by a code number that will be used as a search argument to find the desired entry in the table. The last entry in each FVT has code number 0 and contains either 0 or the address of another FVT. When a task is to be activated, NCP looks in the first FVT for the unique code number that is associated with the task. If the desired code is not found, NCP searches the chained FVTs until the code is found.

Chained FVTs enable the number of external symbol dictionary (ESD) entries and the storage required for the FVTs to be minimized. Therefore, commonly used tasks can be included in one FVT. Tasks that are common to smaller groups of routines can be included in their own FVTs. Tasks that are unique to a resource can be listed in FVTs that are associated only with that resource.

Each task is assigned a unique code between 0 and 255 (codes 3 to 9 are reserved). This code shows which task address is to be inserted in the QCB when a task is to be activated. Use the NCHNG macro to insert the address in the QCB. Table 19 shows the format of an FVT.

Table 19. Format of the Function Vector Table

Code Number	Contents
1	Address of initialization routine
2	Address of notify routine
m	Address of *m*th routine
	⋮
n	Address of *n*th routine
0	Address of next FVT or zero if this is the last FVT

FVTs can be shared by as many resources as desired. When FVTs are chained, the address given with the first occurrence of a specific code number is the one used. FVTs are read-only control blocks and cannot be changed by either NCP or your code. However, the task's pointer in the QCBs can be changed. Whenever a task is changed, the replaced task's FVTABLE code number and the new task's FVTABLE code numbers are stored in the control block for the programmed resource (VLB, NPB, NLB, or NLX).

Examples of FVTs that provide maximum efficiency yet use a minimum of storage are:

- One FVT for all VLBs
- One FVT for all NPBs
- One FVT for all NLBs and NLXs.

Figure 27 is an example of chained FVTs that shows how the code numbers get addresses of the appropriate tasks. Codes 1 and 2 use common tasks but are required in each FVT. The user initialization routine is given control when NCP initialization has been completed. The notify routine is the user notify task.

NCP selects the task address associated with a QCB by beginning with the FVT pointed to by the programmed resource and by scanning the FVT codes for the code matching the code in the QCB. The first matching code found is accepted. In Figure 27, FVT2 differs from FVT1 in the following ways:

- FVT2 has additional tasks (codes 14 and 19).
- FVT2 executes a different awaiting-activation routine for code 17.
- FVT2 executes a different function for code 21.

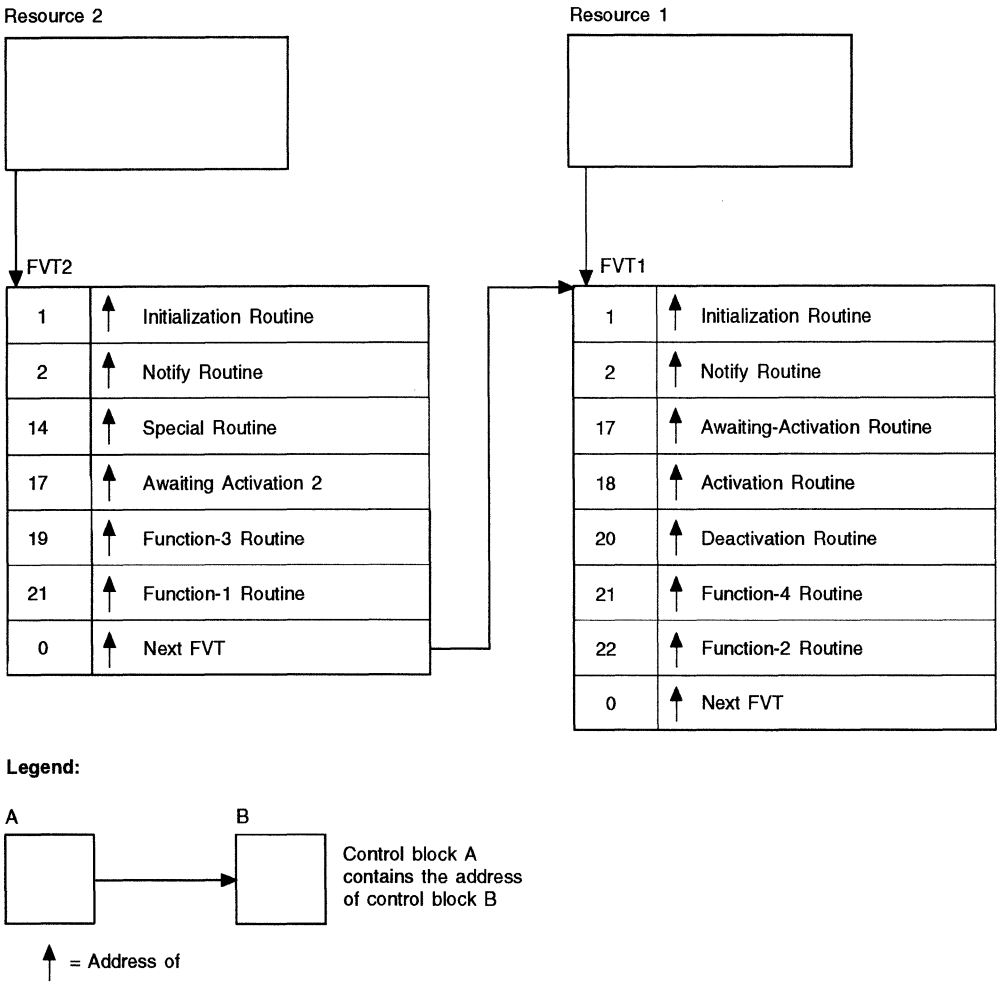


Figure 27. Chained FVTs. FVT1 and FVT2 share tasks for codes 1, 2, 18, 20, and 22.

Figure 28 shows an example of FVT chaining and sharing.

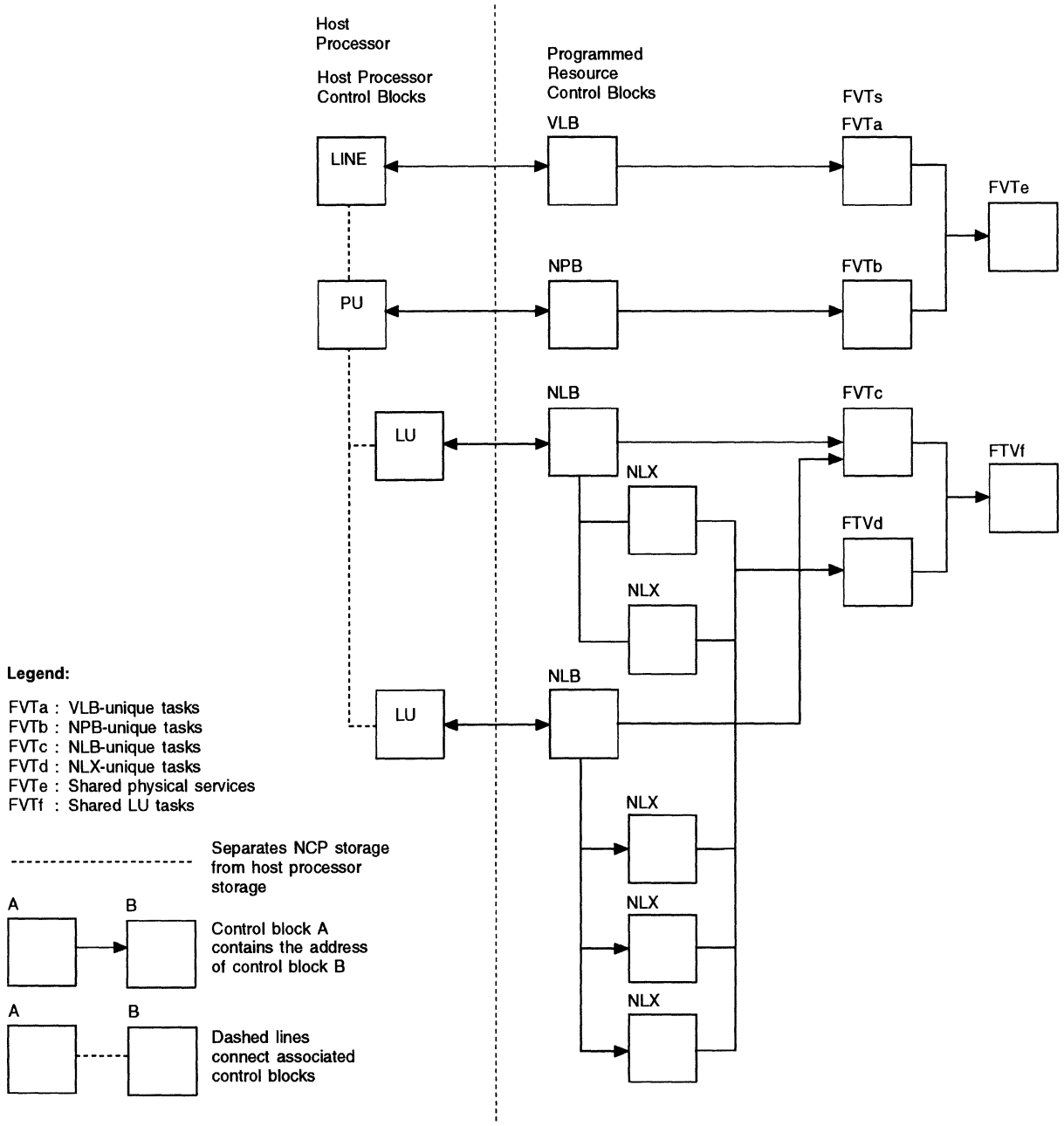


Figure 28. Chaining and Sharing FVTs. The left-hand column contains host-processor control blocks. The center of the figure has control blocks for the programmed resources. The right-hand side of the figure shows the FVTs.

Routing PIUs to Another Network Address (NEOXPORT Macro)

In coding routines to handle programmed resources, use the NEOXPORT macro in NCP coding to route PIUs to the network address that is specified in the DAF field of the PIU.

Obtaining Information from Control Blocks for Programmed Resources (NPARMS Macro)

Use the NPARMS macro to get information from the control blocks that are associated with programmed resources. This macro insulates you from changes in the control blocks. There are two possible expansions of the NPARMS macro. If the CBTYPE keyword is coded, the macro generates in-line code; otherwise, the macro generates a parameter area and a call to the CXDNPRM service subroutine. The in-line expansion provides more efficient performance but can require more storage if it is used many times. Also, the in-line expansion does not verify the control block but depends on proper specification of the CBTYPE keyword. The expansion that makes a call to the subroutine, on the other hand, does not require that you know the type of control block. This expansion verifies that the request is authentic for the control block at the address provided.

The NPARMS macro uses registers 1 and 6 as work registers, so the contents of these registers are not preserved over the macro call. The NPARMS macro without the CBTYPE keyword can be used only in a routine that runs under storage-protection key 1. When the NPARMS macro is coded with the CBTYPE keyword, the macro can be used in a routine that runs under any storage-protection key. This combination can also be used in a routine that runs before storage-protection keys are set.

NCP passes the address of the control block being serviced (VLB, NPB, NLB, or NLX) to the user task in register 2. That address is placed in the register specified by the CNTLBLK keyword. Up to five keywords are coded to specify the information desired and the registers where the information is to be returned.

If the information requested is contained in a field that is less than a full register, the information is right-justified in the return register. When the EXTENSC or EXTENSS keywords are specified, the return register must contain parameter information. If the requested information cannot be obtained, the return register contains 0. If the specified control block is not a valid control block for programmed resources, or if a register other than registers 2 through 7 (except 6) is used to specify the control block, register 1 contains 0 on return to the user routine. When register 1 contains 0, you must assume that none of your requests for information were satisfied.

Changing Fields in Control Blocks for Programmed Resources (NCHNG Macro)

Use the NCHNG macro to change fields in a VLB, NPB, NLB, NLX, or NQE control block. This macro is supplied because the control blocks are located in protected storage and cannot be changed directly from user code. The macro also protects the user as much as possible from future changes in the format of the control blocks. The NCHNG macro generates a parameter-list area and an SVC call to the NCHNG service routine. Do not specify register 1 in a macro operand because the macro service routine uses it to pass its parameter-list area across the supervisor call and to return error information. The NCHNG macro also sets and resets the notify-immediate bit. This bit controls the NOTIFY options of the NCPNAU, PU, and LU definition statements during NCP generation.

If the changes requested by the macro can be made, register 1 contains 0 on return to the user program. If any of the requested changes cannot be made, the low-order byte of register 1 is used to return a code that identifies the error or errors. Table 20 lists the possible return codes from the NCHNG macro.

Table 20. Return Codes from the NCHNG Macro

Bit	Error
0	Specified argument not found in FVTABLE
1	Request not valid for the specified control block
2	New session partner already in session
3	Specified NLX not available for a new session

Returning from a Timer Routine (TVSRTRN Macro)

Use the TVSRTRN macro in your timer routine to generate a proper return linkage to NCP. This macro insulates you from the timer service routines. The macro must be issued with register 6 containing the same data that it contained when your timer routine received control. Register 6 contains the address of a save area and must not be modified. If you need a save area, use the SAVE and RESTORE macros to get to and from the next save area in the chain.

Passing a PIU to NCP Physical Services (NEOENQ Macro)

Use the NEOENQ macro to pass to NCP physical services any PIU that must be passed to the NCP physical services-SSCP session. Physical services sends the PIU using the DAF already in the PIU, but with the OAF and sequence control of NCP physical services.

Changing a Request PIU into a Response PIU (EXCR Macro)

Use the EXCR macro to change a request PIU into a response PIU and to either issue an XPORT macro to send the PIU to the proper NCP element, or return the PIU to your routine, depending on the options specified.

Associating a Session with a Virtual Route (ATTACHVR Macro)

Use the ATTACHVR macro to associate a session with a virtual route. This macro chains the resource connection control block (RCB) of the resource associated with the session to the virtual resource table for the virtual route.

A suggested sequence for associating a session with a virtual route follows:

1. Issue an ACTVRIT macro. This activates the virtual route and places the virtual route vector table (VVT) index for the virtual route into the virtual resource table.
2. Issue an ATTACHVR macro to associate the session with the virtual route.

Note: When you issue the ACTVRIT macro, you can get a return code indicating that the virtual route is already active. This does not prevent a second task from using the virtual route, but a DACTVRIT macro issued by the task that activated the virtual route will also deactivate the virtual route for the second task. Users are restricted to virtual routes 0,0 through 5,2. Sharing and control within that range is a user responsibility.

Terminating the Association of a Session with a Virtual Route (DETACHVR Macro)

Use the DETACHVR macro to terminate the association of a session with a virtual route. The following sequence is suggested:

1. Issue a DETACHVR macro to terminate the association between the resource control block for the resource associated with the session and the virtual resource table for the virtual route.
2. Issue a DACTVRIT macro to deactivate the virtual route.

   Note: If a second task is also using this virtual route, the DACTVRIT macro also deactivates the virtual route for the second task.

Checking the Status of a Virtual Route (CHECKVR Macro)

Use the CHECKVR macro to check the state of a virtual route. The virtual route state is indicated in a specified register as being either held or not held. The held state does not allow a request for data, whereas the not-held state allows the request for data to continue.

The CHECKVR macro returns the flag field of the requested VRB. The bit definitions of the flag field are listed in the description of the VRB in *NCP and EP Reference Summary and Data Areas*, Volume 1.

Notification of Virtual Route Status Change

During NCP generation, you can specify a routine to be executed if the virtual route status changes. The status can go from held to not held or from not held to held. The routine status is passed in register 1. Register 1 contains a 0 if the virtual route status is in the held state, and contains a nonzero value if it is in the not-held state. The held state does not allow the requesting of data, whereas the not-held state allows the request for data to continue. See *NCP and EP Reference* for details on virtual route pacing.

Obtaining a Pointer to the Next Resource Control Block in the VRB Chain (RCBSCAN Macro)

Use the RCBSCAN macro to get the address of the next RCB queued on the VRB. The type of resource associated with the RCB is also returned.

Reserving a Buffer Before the Buffer Is Needed (PRELEASE Macro)

Reserve one or more buffers for future use with the PRELEASE macro. If the requested buffers are not available, the requesting program can be posted until the buffers are available. Other level 5 tasks continue. When the reported number of buffers becomes available, the task is dispatched. You can use this macro to help regulate network flow.

If buffers have been reserved and are not used by SYSXIT time, they are reassigned to the buffer pool.

Note: A PRELEASE macro does not lease buffers. A LEASE macro function must still be performed.

Using NPM Data Collection with Programmed Resources

You can use the NetView Performance Monitor (NPM) performance management function with NCP to accumulate network data about user-written programmed resources. The NPM function collects the data, logs it, and when specified, displays the data at an operator station.

NCP provides an interface that allows NPM to collect performance statistics for programmed resources. This interface allows NCP to communicate NPM-NCP session data collection activity to the user-written code. NCP issues NPM activity notifications to the user-written code indicating NPM requests to start or stop collecting statistics on one of its resources. If specified by IBM special products or user-written code, NCP also communicates NPM FORWARD requests. NCP notifies the user-written code when to supply its statistics when building the COLLECT RU. NCP maintains the COLLECT RUs, but each IBM special products or user-written code resource must build its own resource record in the RU.

Preparing the Programmed Resource for Collection

NCP uses the FVTABLE macro to communicate with user-written code running in level 5.

To process NPM data collection requests for user-written code, code the USRNPM and COLNPM keywords on the FVTABLE macro. Table 21 summarizes these keywords.

Table 21. FVTABLE Keyword Descriptions

Keyword	Offset into the FVTABLE	Definition
USRNPM	X'0C'	User task begun for NPM Start, Stop, or Forward notification (address)
COLNPM	X'10'	User collect routine used for NPM data collection (address)

The user task is a level 5 task, given control by the supervisor. The task is begun when a START RU or STOP RU that contains a resource address defined for the user-written code is received. The user task is begun for a FORWARD RU only when an NCHNG, requesting notification of forward processing, is issued by the user-written code. See page 66 for a description of the NCHNG macro.

NCP branches to the routine in the FVTABLE for collection of data. This is a level 5 routine, called directly by NCP to allow the user-written code to copy the data for its resource into the COLLECT RU.

The user task can be invoked for Start, Stop, or Forward notification according to the task modifier in the resource control block of the user-written code. VLBNPMIF is the task modifier for a virtual line; NPBNPMIF is the task modifier for a virtual physical unit (PU). For both the VLB and the NPB, the offset is X'18' and is 1 byte long. Table 22 on page 100 describes the the NPM task modifiers.

Table 22. NPM Task Modifier Descriptions

Name	Value	Description
NPM START	X'01'	Indicates a START request received for the resource
NPM FORWARD	X'02'	Indicates a FORWARD request received for the resource
NPM STOP	X'03'	Indicates a STOP request received for the resource

Start Processing for Programmed Resources

When an NCP receives a START RU, NCP verifies its correctness, including validating each resource in the START RU. For data collection to begin on a specific resource, the resource must have been defined in the NCP generation as eligible for NPM performance data collection. NCP verifies the NPM task and routine entries in the FVTABLE to be nonzero. If either entry is a zero, NCP returns a negative response to NPM.

When eligibility is established, start processing begins on the resource. NCP retrieves an NQE from the NQE pool and formats it for the resource. NCP sets up the element address and type byte.

The programmed resource receives control from NCP when the task setup in the FVTABLE is triggered for NPM activity. When NCP receives a START RU from NPM, NCP begins the FVTABLE NPM task with the Start modifier. The user-written code begins start processing when dispatched by the supervisor. Use the NCHNG macro to set the NQE resource record length of the resource record and to request notification by NCP when a FORWARD RU is received for the programmed resource.

Forward Processing for Programmed Resources

When a FORWARD RU is received from NPM, NCP builds a COLLECT RU header to begin collect processing. It determines which resources to collect data from: IBM special products or user-written code, NCP, or both. For user resources that request Forward notification, NCP begins the user NPM task with the Forward modifier. When all of the resources to be collected are considered for notification, NCP begins the NPM Collect and exits level 5, allowing programmed resources to do forward processing.

Stop Processing for Programmed Resources

When NCP receives a STOP RU from NPM, NCP begins the FVTABLE NPM task with the Stop modifier. NCP returns the NQE to the NQE pool and responds positively to the STOP PIU.

Collect Processing for Programmed Resources

NCP collects performance data for NCP resources. After data collection for all NCP-owned resources has been completed and a 2-second interval has expired, NCP performs data collection for all the user-written code resources. NCP collects data for user-written code resources by branching into the user collect routine specified in the FVTABLE.

When entered, the user-written code resources must copy their data into the COLLECT RU. This data is the total resource record defined by NPM excluding the first 4 bytes. These 4 bytes contain the element address (2 bytes), the type byte (1 byte), and the length of the resource record (1 byte).

Note: The length of the resource record is the total length of the resource record including the 4 bytes that NCP sets up.

User-written code receives control from NCP in level 5. The input and output associated with this interface are described in Table 23. Figure 29 on page 102 illustrates the relationship of the control blocks and buffers passed using the Register 1 parameter area.

Table 23. Interface for NCP to User-Written Code Collect Routine

Input or Output	Parameter Passed	Description
Input	Register 1	Pointer to the parameter area
Input	Parameter area fullword 1	Pointer to the VLB if collecting on a virtual line; pointer to the NPB if collecting on a virtual PU
Input	Parameter area fullword 2	Pointer to the NQE associated with the resource
Input	NQE offset X'10'	Pointer to the buffer currently being written into
Input	Register 6	Pointer to the NCP save area
Input	Register 7	Return address to NCP when collection is complete
Output	Register 1	Return code; nonzero is an error
Output	Register 6	Pointer to save area originally given to the user-written code routine

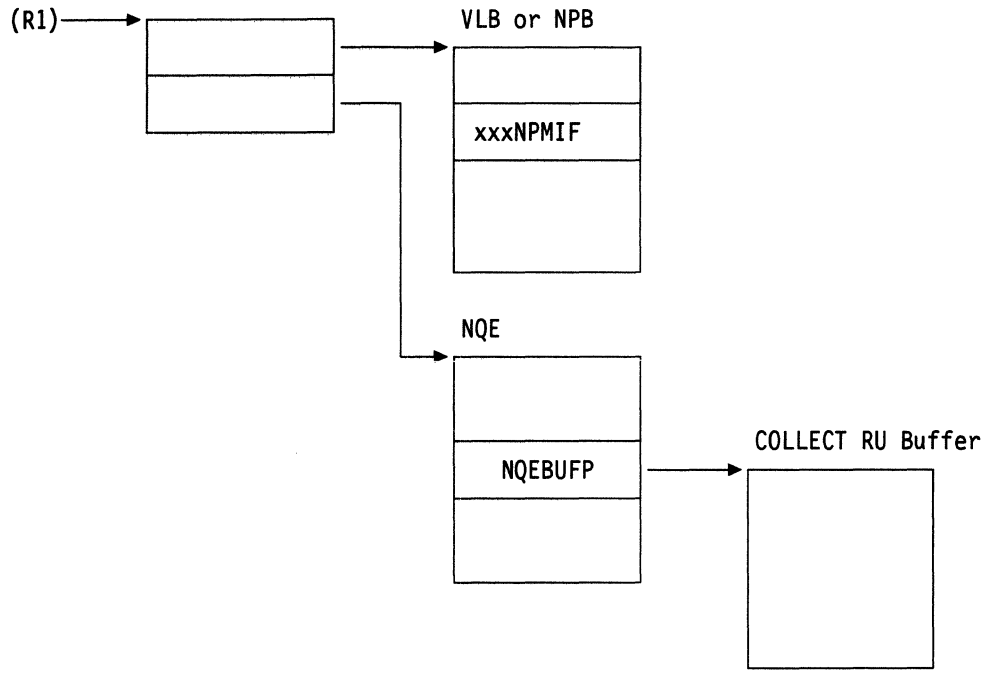


Figure 29. Relationship between Control Blocks and Buffers

When it receives control, the user-written code locates the current COLLECT RU buffer in the NQEBUFP field of the NQE. This buffer may already contain data, so you must use the DATCNT and OFFSET fields to find the next available byte. If the data will not fit into one buffer, you must span the data to the next buffer provided in the chain. NCP calculates the buffer requirement based on the length of the NQE set up by user-written code during start processing.

NCP expects the user-written code resource to update the buffer count in each buffer into which it writes. NCP validates the counts, updates the TH count, and increments the number of resource records in this COLLECT RU. If the counts calculated by the user-written code and NCP do not match, NCP indicates that the collection failed by sending NPM only the 4-byte header, which NCP set up.

NCP examines register 1 for a return code. If register 1 is nonzero, NCP sets the record length to 4 and sends only the record header to NPM.

Using NPM Session Accounting with Programmed Resources

User-written programs usually support devices that NCP does not support. The NCP definition statements that define these devices may mean something entirely different to the user-written code. For example, a group of SDLC link and associated PU and LU definition statements may represent a virtual circuit to NCP Packet Switching Interface (NPSI). You may want to capture accounting statistics specific to the protocols your user-written code uses.

You must activate the SNA representation for any user-written code intended for communication with NCP. The user-written code gets control from NCP during the activation sequences to provide the activities necessary to activate the resource.

An SNA-level session may coexist with a protocol-level interaction supported by the user-written code. If both NCP and the user-written code are performing accounting activities, NPM could receive accounting records from both sources describing the same transaction.

Use the NPAQINFO macro to allow an IBM special products or user-written code program to acquire session information for an SNA LU-LU. This macro lets the IBM special products or user-written code include the session information in its accounting records. The IBM special products or user-written code program can obtain the following types of information:

- Logical unit name
- Control point name
- Subarea address of logical unit
- Subarea address of session partner logical unit
- Element address of logical unit
- Element address of session partner logical unit
- Network ID of logical unit
- Network ID of session partner logical unit
- Network ID of control point
- Procedure correlation ID (PCID).

When you use this macro, you must specify the address of an LU control block. When the macro is executed, it determines if NCP includes the boundary session accounting function. If accounting is not included, a return code indicating that the function is not supported is set. If accounting is supported, it checks to determine if the input logical unit is involved in a session NCP is collecting accounting data for. If the logical unit is not involved, a return code is set indicating this.

Part 2. SSP Customization

Chapter 4. Creating and Using User-Written Generation Applications

This chapter contains General-Use Programming Interface and Associated Guidance Information.

User-written generation applications customize the way the NCP generation definition is processed. Generation applications are integrated into the generation process through the NCP/EP definition facility (NDF) standard attachment facility. The standard attachment facility consists primarily of linkage conventions and NDF routines your applications can use.

Your applications can process NCP statements and keywords, as well as user-defined statements and keywords, in an NCP generation definition. The applications can use NDF internal utilities to aid in processing the generation definition. For information on what routines these NDF utilities perform, see "What the Utilities Can Do" on page 123.

A generation application consists of an executable load module that NDF loads and calls during generation. You can write your application in any language that provides executable code that interfaces properly with NDF routines. NDF routines use standard IBM assembler language interface conventions. All code examples in this book are written in IBM 370 Assembler Language.

What Generation Applications Do

A generation application processes statements and keywords during the NDF phase of the generation process. You might want to use generation applications for the following purposes:

- Generate the link-edit statements needed to include user-written code in NCP.
- Provide your own defaults for standard macros and keywords. For example, there may be values for keywords that don't normally have defaults that are almost always the same in your network.
- Provide your own macros and keywords. For example, you might want to provide statements and keywords that control the generation of control blocks for user-written NCP code.

For each statement, you create routines in your application that handle various processing steps: statement prolog routines, keyword routines, and statement epilog routines. The prolog and epilog routines handle the housekeeping chores of the processing tasks such as checking statement sequence and initializing variables. The keyword routines for a statement handle tasks like verifying and substituting keyword values. You can have as many prolog, epilog, and keyword routines as you need for each statement.

Structure of a User-Written Generation Application

A generation application consists of two main parts: the routines and the statement/keyword vector table (SKVT). The routines handle the various processing tasks for the application, and the SKVT indicates to NDF which statements and keywords your application should process.

Your generation application's routines should contain their routine names five bytes past their entry points. NDF will use bytes 6 through 13 for the routine name when printing a procedure traceback, for instance, when a severity-12 error message is issued.

The generation application lists routines for each statement in its SKVT. When NDF sees that your application should process a statement, it uses the locations in the SKVT to call the routines for that statement. NDF also passes certain parameters to your routines depending on the routine type. The next section explains these parameters.

How NDF Invokes User Routines

When NDF calls your generation application's routines, it uses standard IBM 370 Assembler Language linkage conventions to pass a pointer to the parameter list. The following parameter descriptions give the lengths of each parameter. The sample generation application on page 125 shows how to use the NDF parameters.

Each of the routine types (prolog, epilog, and keyword) has its own parameter list. Each parameter list includes the NDF status word and the transfer vector table (XVT). The NDF status word indicates the states of various traces and other options. The status word is described under "Format of the NDF Status Word" on page 110. The XVT is a list of pointers to the NDF internal utilities. The XVT is described under "Using the Transfer Vector Table" on page 123.

Parameters of the Statement Prolog Routine

NDF passes the following parameters to a statement prolog routine:

Status

The NDF status word.

Statement name

The name of the statement being processed, such as `BUILD%`, represented as a 16-character string in NDF string standard representation.

Note: NDF uses its own string-handling system called *string standard representation*. String standard representation is explained in "NDF String Handling" on page 120.

Prolog

The constant `PROLOG%` as a 16-character string in NDF string standard representation.

Dispatch flag

When set to 1, a flag that indicates to NDF that the rest of the routines defined for this statement in the generation application's SKVT are not to be dispatched. The flag is a single byte.

XVT
The transfer vector table.

Version and Release
The version and release of NDF in the format `V3R2%` as a 16-character string in NDF string standard representation.

Parameters of the Statement Epilog Routine

NDF passes the following parameters to a statement epilog routine:

Status
The NDF status word.

Statement name
The name of the statement being processed, such as `BUILD%`, represented as a 16-character string in NDF string standard representation.

Epilog
The constant `EPILOG%` as a 16-character string in NDF string standard representation.

XVT
The transfer vector table.

Parameters of the Keyword Routine

NDF passes the following parameters to a keyword routine:

Status
The NDF status word.

Statement name
The name of the statement being processed, such as `BUILD%`, represented as a 16-character string in NDF string standard representation.

Keyword name
The name of the keyword being processed, such as `MODEL%`. If you are processing a statement symbol, the keyword name is `#SYMBOL%`. The keyword name is represented as a 16-character string in NDF string standard representation.

Keyword value
The user-coded value for the keyword. If the keyword has not been coded, the value is null (%). It is represented as a 256-character string in NDF string standard representation. It is passed exactly as it is coded in the generation definition.

Value length
The logical length, in characters, of the keyword value. It is a fullword integer and has a value of 0 if the keyword has not been coded.

Parsed value
A value that points to a parsed representation of the keyword value if the keyword value is coded as a value list.

A parsed value list is an array consisting of the logical length of each subvalue string and a pointer to each subvalue string. The string values themselves are held elsewhere in storage. The subvalue length

and pointer fields are both 4-byte fields. The following example shows how the array is represented in memory:

```
Length of subvalue 1
Pointer to subvalue 1
Length of subvalue 2
Pointer to subvalue 2

:
Length of subvalue n
Pointer to subvalue n
```

From the keyword value, NDF generates the subvalue strings in the parsed value list. It does not include the enclosing parentheses, if any are coded, and it removes the commas separating the subvalues of a keyword.

Note: If a keyword has only one subvalue and the subvalue is not enclosed in parentheses, then the subvalue and keyword value are the same.

For example, the subvalues of the keyword value `(sub1,,(sub3))` would be stored as three strings: `sub1%`, `%`, and `(sub3)%`.

Parsed values count
The number of parsed values for this keyword. It is a fullword positive integer.

XVT
The transfer vector table.

Format of the NDF Status Word

All routines called by NDF have the NDF status word as the first parameter. The status word is a 32-bit string. Note that user generation applications cannot directly set the bits of the NDF status word. Table 24 explains the bits of the NDF status word.

Table 24 (Page 1 of 2). NDF Status Word Bits

Bit	Explanation
0	Set to 1 if TRPARM=SK is coded on an OPTIONS statement. The bit is reset to 0 if NOTRPARM=SK is coded.
1	Set to 1 if TRPROC=SK is coded on an OPTIONS statement. The bit is reset to 0 if NOTRPROC=SK is coded.
2	Internal to NDF.
3	When set to 1, NDF suppresses table assembly source production. The bit is set because either FASTRUN=ON was coded or an error has occurred.
4	Set to 1 if TYPSYS=DOS is coded on the BUILD statement.
5	Set to 1 if NDF is now punching source for the TABLE 2 assembly.
6	Internal to NDF.
7	Internal to NDF.
8	Set to 1 if FASTRUN is coded on the OPTIONS statement.
9	Internal to NDF.
10	Internal to NDF.

Table 24 (Page 2 of 2). NDF Status Word Bits

Bit	Explanation
11	Internal to NDF.
12	When set to 1, each routine is traced as it is dispatched.
13	Internal to NDF.
14	Internal to NDF.
15	Internal to NDF.
16	Set to 1 if a warning (severity 4) message has been issued.
17	Set to 1 if NEWDEFN=YES was specified on an OPTIONS statement that is also the first statement in the generation definition.
18	Set to 1 if the current routine is performing post-processing for the current keyword. The bit is set to 0 otherwise.
19	Internal to NDF.
20	Internal to NDF.
21	Set to 1 if NEWDEFN=REUSE was specified on an OPTIONS statement that is also the first statement in the generation definition. This allows the NEWDEFN file to be submitted to NDF.
22-31	Reserved. The value of these bits is undependable.

Making a Statement/Keyword Vector Table

NDF uses the statement/keyword vector table (SKVT) for your application to determine when to give control to that application. The SKVT must begin at the entry point of your generation application.

The SKVT contains records that define the statements and keywords that your generation application will process. The first record in the SKVT must be a START record.

Prolog, epilog, and keyword records follow each statement record. The prolog and epilog records define routines that NDF calls every time the statement is coded.

Each keyword record defines a keyword and a routine that validates the keyword. The keyword record also indicates whether or not the keyword routine is to be called every time the statement is coded or only when the keyword is coded on the statement.

NDF calls all prolog, epilog, and keyword routines in the order in which they occur in the SKVT. The keywords are not processed in the order in which they are coded.

The SKVT records and their unique type numbers are:

1. Start
2. Statement
3. Prolog
4. Epilog
5. Keyword and null keyword
6. Continue
7. End.

Format of the Statement/Keyword Vector Table

Figure 30 shows the format of the SKVT records. Each record is 20 bytes long. For each statement to be processed by your application, you must have a start record as the first record for that statement and an end record as the last record for that statement. The other records can be in any order, and you can have more than one of each type of record.

The next section gives sample macros you can use to create SKVT entries. See the sample generation application on page 125 for an example of how these macros are used.

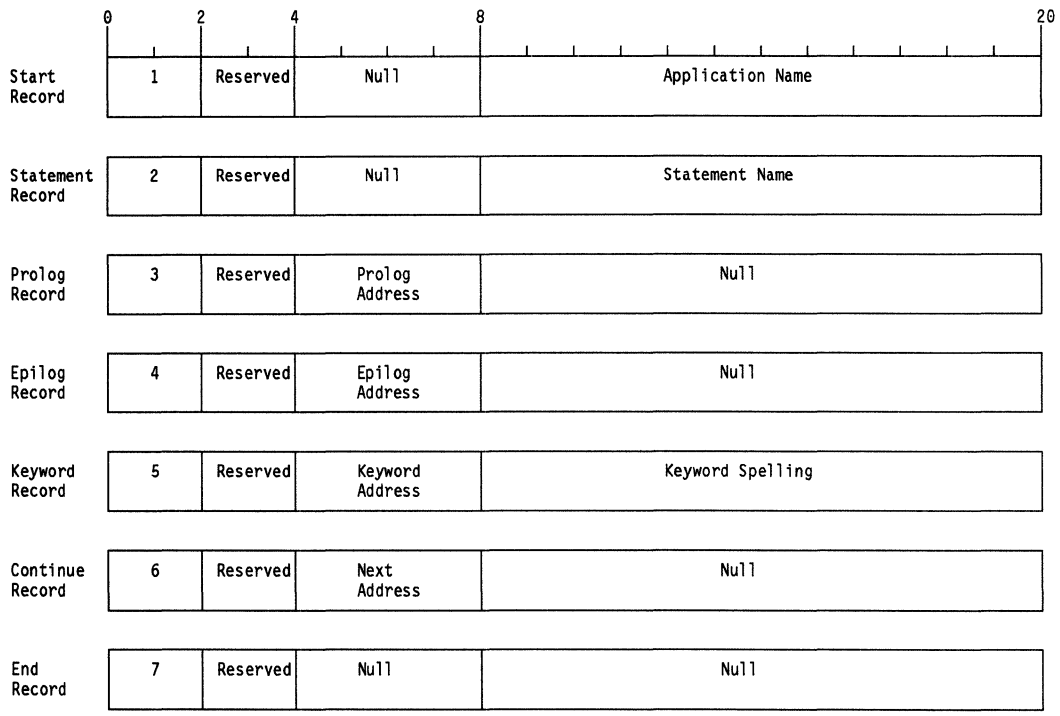


Figure 30. Format of the SKVT

Definition Statements for Generating SKVT Records

The following macros generate the various types of SKVT records. You can code these macros yourself as shown here, or you can create equivalent routines.

Start Record Macro

The start record macro contains the name of the SKVT. It also indicates to NDF whether the generation load module name coded on the USERGEN macro can be used on the LNKOWNER or VIROWNER keywords. The start record must be the first record in the SKVT entry for this routine.

The LNKOWN and VIROWN parameters set bits that indicate whether or not the application name is valid on the LNKOWNER and VIROWNER keywords.

Figure 31 shows the start record macro.

```
* Start record macro
     MACRO
&X     ICNVTSTR &NAME=,&LNKOWN=,&VIROWN=
       DC      XL2'1'                  Record type
       AIF     ('&LNKOWN' EQ 'YES').LNKYES
       AIF     ('&VIROWN' EQ 'YES').VIRYES
       DC      XL2'0'                  Neither VIROWNER or LNKOWNER
       AGO .STRTRST
.VIRYES ANOP
       DC      XL2'1'                  Valid on VIROWNER
       AGO .STRTRST
.LNKYES ANOP
       AIF     ('&VIROWN' EQ 'YES').VIRLNK
       DC      XL2'2'                  Valid on LNKOWNER
       AGO .STRTRST
.VIRLNK ANOP
       DC      XL2'3'                  Valid on both VIROWN and LNKOWN
.STRTRST ANOP
       DC      XL4'0'                  Null address
       DC      CL12'&NAME'             Routine name
     MEND
```

Figure 31. Example of a Start Record Macro

Statement Record Macro

The statement record macro contains the name of either a user-defined or an NCP macro. The statement record defines the statement to which subsequent prolog, epilog, and keyword macros apply. The SKVT section for this statement is ended by an end record or by another statement record.

Note: Any keywords you define should be defined with a three-letter prefix unique to the generation application; for example, the USR.NEWKWD keyword with USR as the unique prefix. If you are a VTAM user, you must do this to prevent errors when VTAM activates your NCP.

Figure 32 shows the statement record macro.

```
      MACRO
&NAME  ICNVTSTM &PROC=(1,0)
*    Generate statement name record
.* Input:  PROC=positional parameter indicating that pre- and/or post-NCP
.*         processing routines are associated with this statement.
.*         (1,0)=pre, (0,1)=post, (1,1)=pre and post
.*
        DC      XL2'2'              Record type
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' NE '1').SPRE
        AIF     ('&PROC(1)' NE '1' AND '&PROC(2)' EQ '1').SPOST
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' EQ '1').SBOTH
.SPRE   ANOP
        DC      XL2'0000'           Reserved
        AGO     .SNAME
.SPOST  ANOP
        DC      XL2'C000'           Reserved
        AGO     .SNAME
.SBOTH  ANOP
        DC      XL2'4000'           Reserved
.SNAME  ANOP
        DC      XL4'0'              Reserved
        DC      CL12'&NAME'         Statement name
      MEND
```

Figure 32. Example of a Statement Record Macro

Prolog Record Macro

The prolog record macro defines a routine that NDF calls for the current statement. A statement can have any number of prolog routines. NDF calls these routines with the prolog parameter list.

Note: You should process the prolog routines at the beginning of a statement. The pre- and post-indicators in the statement record macro are used by NDF for initialization and must indicate the type of processing needed by the SKVT records following this statement record.

Figure 33 shows the prolog record macro.

```
      MACRO
&X      ICNVTPRO &PROLOG=0,&PROC=(1,0)
*       Generate prolog record
.* Input:  PROLOG=prolog routine name
.*         PROC=positional parameter indicating that pre- and/or post-NCP
.*         processing routines are associated with this statement.
.*         (1,0)=pre, (0,1)=post, (1,1)=pre and post
.*
        DC      XL2'3'              Record type
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' NE '1').PPRE
        AIF     ('&PROC(1)' NE '1' AND '&PROC(2)' EQ '1').PPOST
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' EQ '1').PBOTH
.PPRE   ANOP
        DC      XL2'1000'           Reserved
        AGO     .PRADDR
.PPOST  ANOP
        DC      XL2'4000'           Reserved
        AGO     .PRADDR
.PBOTH  ANOP
        DC      XL2'0'              Reserved
.PRADDR ANOP
        AIF     ('&PROLOG' EQ '0').NPROL
        DC      V(&PROLOG)          Prolog routine address
        AGO     .CPROL
.NPROL  ANOP
        DC      XL4'0'              Null address
.CPROL  ANOP
        DC      CL12' '             Null spelling
      MEND
```

Figure 33. Example of a Prolog Record Macro

Epilog Record Macro

The epilog record macro defines a routine that NDF calls for a statement using the epilog parameter list. NDF calls each epilog routine when its statement is coded. A statement can have any number of epilog routines.

Note: You should process epilog routines at the end of a statement.

Figure 34 shows the epilog record macro.

```
      MACRO
&X      ICNVTEPI &EPILOG=0,&PROC=(1,0)
.* Input:  EPILOG=epilog routine name.
.*         PROC=positional parameter indicating that pre- and/or post-NCP
.*         processing routines are associated with this statement.
.*         (1,0)=pre, (0,1)=post, (1,1)=pre and post
.*
        DC      XL2'4'              Record type
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' NE '1').EPRE
        AIF     ('&PROC(1)' NE '1' AND '&PROC(2)' EQ '1').EPOST
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' EQ '1').EBOTH
.EPRE   ANOP
        DC      XL2'1000'           Reserved
        AGO     .ERADDR
.EPOST  ANOP
        DC      XL2'4000'           Reserved
        AGO     .ERADDR
.EBOTH  ANOP
        DC      XL2'0'              Reserved
.ERADDR ANOP
        AIF     ('&EPILOG' EQ '0').NEPIL
        DC      V(&EPILOG)          Epilog routine address
        AGO     .CEPIL
.NEPIL ANOP
        DC      XL4'0'              Null address
.CEPIL ANOP
        DC      CL12' '             Null spelling
      MEND
```

Figure 34. Example of an Epilog Record Macro

Keyword Record Macro

The keyword record macro names the keyword to be processed and the routine needed to do the processing. The keyword can be either a user-defined or an NCP keyword.

The first bit in the flag field of the keyword record macro indicates whether the keyword routine will be called even if the keyword was not coded on the macro. If the first bit is set to 1, the keyword routine is called only when the keyword is coded. If the first bit is set to 0, the keyword routine is called whether or not the keyword is coded.

Figure 35 on page 118 shows the keyword record macro.

```
      MACRO
&X      ICNVTKEY &ROUTINE=0,&IFFCODE=(NO,NO),&NAME=,&PROC=(1,0)
.*         Generate keyword record
.* Input:  ROUTINE=keyword routine name.
.*         IFFCODE=flag if routine is to be called even if the associated
.*         keyword is not coded.  Positional subvalues are
.*         the same as for the PROC parameter.
.*         NAME=keyword name.
.*         PROC=positional parameter indicating that pre-and/or post-NCP
.*         processing routines are associated with this statement.
.*         (1,0)=pre, (0,1)=post, (1,1)=pre and post
.*
        DC      XL2'5'              Record type
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' NE '1').KPRE
        AIF     ('&PROC(1)' NE '1' AND '&PROC(2)' EQ '1').KPOST
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' EQ '1').KBOTH
.KPRE   ANOP
        AIF     ('&IFFCODE(1)' EQ 'YES').PREYES
        DC      XL2'1000'           Reserved
        AGO     .KRADDR
.PREYES ANOP
        DC      XL2'9000'
        AGO     .KRADDR
.KPOST  ANOP
        AIF     ('&IFFCODE(2)' EQ 'YES').POSTYES
        DC      XL2'4000'           Reserved
        AGO     .KRADDR
.POSTYES ANOP
        DC      XL2'6000'           Reserved
        AGO     .KRADDR
.KBOTH
        ANOP
        AIF  ('&IFFCODE(1)' EQ 'YES' AND '&IFFCODE(2)' EQ 'YES').YY
        AIF  ('&IFFCODE(1)' NE 'YES' AND '&IFFCODE(2)' EQ 'YES').NY
        AIF  ('&IFFCODE(1)' EQ 'YES' AND '&IFFCODE(2)' NE 'YES').YN
        AIF  ('&IFFCODE(1)' NE 'YES' AND '&IFFCODE(2)' NE 'YES').NN
.NN     ANOP
        DC      XL2'0'              Reserved
        AGO     .KRADDR
.YN     ANOP
        DC      XL2'8000'           Reserved
        AGO     .KRADDR
.NY     ANOP
        DC      XL2'2000'           Reserved
        AGO     .KRADDR
.YY     ANOP
        DC      XL2'A000'           Reserved
.KRADDR ANOP
        DC      V(&ROUTINE)         Keyword processing routine address
        DC      CL12'&NAME'         Keyword spelling
      MEND
```

Figure 35. Example of a Keyword Record Macro

Null Keyword Record Macro

The null keyword record macro indicates a keyword name that is not going to be processed, but that NDF should not flag as unsupported.

Figure 36 shows the null keyword record macro.

```
      MACRO
&X      ICNVTDMK &NAME=,&PROC=(1,0)
*       Generate dummy keyword record
.* Input:  NAME=keyword name.
.*         PROC=positional parameter indicating that pre- and/or post-NCP
.*         processing routines are associated with this statement.
.*         (1,0)=pre, (0,1)=post, (1,1)=pre and post
.*
        DC      XL2'5'              Record type
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' NE '1').DPRE
        AIF     ('&PROC(1)' NE '1' AND '&PROC(2)' EQ '1').DPOST
        AIF     ('&PROC(1)' EQ '1' AND '&PROC(2)' EQ '1').DBOTH
.DPRE   ANOP
        DC      XL2'1000'           Reserved
        AGO     .DRADDR
.DPOST  ANOP
        DC      XL2'4000'           Reserved
        AGO     .DRADDR
.DBOTH  ANOP
        DC      XL2'0'              Reserved
.DRADDR ANOP
        DC      F'0'                Null keyword address
        DC      CL12'&NAME'         Keyword spelling
      MEND
```

Figure 36. Example of a Null Keyword Record Macro

Continue Record Macro

The continue record macro chains SKVTs across CSECTs.

Figure 37 shows the continue record macro.

```
      MACRO
&X      ICNVTCNT &ROUTINE=0
        DC      XL2'6'              Record type
        DC      XL2'0'              Reserved
        DC      V(&ROUTINE)         Continuation address
        DC      CL12' '             Null spelling
      MEND
```

Figure 37. Example of a Continue Record Macro

End Record Macro

The end record macro ends the SKVT.

Figure 38 shows the end record macro.

```
      MACRO
&X      ICNVTEND
        DC      XL2'7'              Record type
        DC      XL2'0'              Reserved
        DC      XL4'0'              Null address
        DC      CL12' '             Null spelling
      MEND
```

Figure 38. Example of an End Record Macro

NDF String Handling

NDF uses its own string format for string manipulation, called *string standard representation.* String standard representation uses special control codes and characters to control string processing. NDF does this by using an argument list for string variables. The first parameter (the control parameter) includes control characters and text. The control characters determine whether the next parameters in the list are included, indicating such things as whether a numeric or string value is expected, whether the numeric value is integer or hexadecimal, and so on. Along with the control codes, NDF uses two special characters to control text manipulation, the escape and end-of-string characters.

Escape and End-of-String Characters

The two characters ¬ and % have special meaning in string standard representation:

¬ The escape character. It removes any special meaning associated with the character that it precedes. This is character X'5F'.

% The end delimiter. It identifies the end of a string in string standard representation. Unless otherwise specified, all strings passed as parameters in NDF are represented using the end delimiter.

Note that the null string in string standard representation is just the end delimiter, %.

Table 25 on page 121 shows strings in string standard representation and their corresponding conventional strings. The logical length of a string in string standard representation is the number of characters in the equivalent conventional string. The actual length of a string in string standard representation is the number of characters in the string, including all escape characters and the end delimiter. The maximum number of characters in the physical representation of a string in string standard representation, including control codes, is 256.

Table 25. Strings in String Standard Representation and Corresponding Conventional Strings

String Standard	Conventional String	Logical Length	Actual Length
'%'	''	0	1
'A/C%'	'A/C'	3	4
'AB/C¬%D%'	'AB/C%D'	6	8
'AB¬¬C¬%D%'	'AB¬C%D'	6	9

NDF provides routines to convert to and from string standard and conventional string format. Refer to *NCP, SSP, and EP Resource Definition Guide* for a description of these routines.

NDF String-Handling Control Codes

Table 26 lists and explains the function of all the NDF string standard control codes. These codes are recognized by NDF string processing utilities (ICNERPST, ICNOBPUN, ICNOBPU2, ICNRPFMT, ICNRPINF, and ICNSMFMT).

Table 26 (Page 1 of 2). NDF String Control Codes

Control Code	Function
/F /*n*F	**Insert a fullword fixed number.** Takes the next argument and inserts it as a fullword integer. If *n* is not specified, the number is printed without leading spaces or zeros. Negative numbers have a leading minus sign. If *n* is specified, the number is right-justified with leading spaces in a field of *n* characters. If the number is too large to fit in the specified field width, the field width is ignored.
/H /*n*H	**Insert a halfword fixed number.** Takes the next argument and inserts it as a halfword integer. If *n* is not specified, the number is printed with no leading spaces or zeros. Negative numbers have a leading minus sign. If *n* is specified, the number is right-justified with leading spaces in a field of *n* characters. If the number is too large to fit in the specified field width, the field width is ignored.
/X /*n*X	**Insert a fullword in hexadecimal.** Takes the next argument and inserts it as a fullword hexadecimal number. Leading zeros are printed. If *n* is omitted, 8 is assumed. If $n \geq 8$, the number is right-justified with leading spaces. If $n \leq 8$, this routine suppresses the first 8 – n digits. If any of these digits are nonzero, a value of *n* is chosen which prints all significant digits.
/B /*n*B	**Insert an n-bit Boolean value.** Takes the next argument and inserts it as an *n*-bit Boolean string. If *n* is omitted, 1 is assumed.
/S /*n*S	**Insert a standard string.** Takes the next argument and inserts it as a string in string standard representation. If *n* is specified, the string is left-justified and padded on the right with blanks in a field of *n* characters. If the string is too large to fit in the specified field width, the field width is ignored.

Table 26 (Page 2 of 2). NDF String Control Codes

Control Code	Function
//S //*n*S	**Insert a standard string and rescan.** Takes the next argument and inserts it as a string in string standard representation. If *n* is specified, the string is left-justified and padded on the right with blanks in a field of *n* characters. If the string is too large to fit in the specified field width, the field width is ignored. The double slash indicates that the string is rescanned, with control codes being interpreted in the same way as in the original input control string.
/*n*T /*n.m*T	**Tab to column n.** Inserts enough space characters to reach column *n*. The column number is initialized to 1 and increased by 1 for every character placed in the output string. If *m* is specified, it indicates the minimum number of spaces that are to be placed in the output string. The default value of *m* is 0. If the current column number is greater than *n* – *m*, then *m* spaces are placed in the output string, even though this makes the column number greater than *n*.

The *n* following the control character generally refers to a field width specification, although the exact meaning depends on the control code it appears with. The *n* must be specified as an unsigned decimal integer or as the letter *V*. If *V* is used, the next argument in the argument list, which must be a 31-bit number, is used to obtain the actual value. If this argument is negative, 0 is assumed.

When no field width is specified, the routines use a field large enough to contain the data. If a field size is specified that is too small to contain the data, the routines ignore it and select a field width of the appropriate size.

The control codes in the input string must correspond to the types of arguments in the argument list. For example, a */F* control code requires a corresponding numeric argument. If the control code and its associated argument do not match, then the results are unpredictable.

Note: NDF cannot determine the actual type of an input argument.

See *NCP, SSP, and EP Resource Definition Guide* for examples of how each routine uses the control codes.

Using NDF Internal Utilities

NDF provides a variety of internal utility routines that your generation application can use to perform various functions during generation. These routines provide most of the functions needed for statement and keyword processing and the creation of an NCP controller load module.

Calling NDF Internal Utilities

When you call an NDF internal utility, you use standard IBM assembler linkage protocols. First set up the linkage registers as follows:

- Put the address of the input and output parameter list in register 1.
- Put the address of your register save area in register 13.

Then issue a BALR instruction to the location supplied in the transfer vector table (XVT). Have the BALR instruction put the return address in register 14.

The following example shows the format used to call NDF routines. The routine name, ICNOBPUN, is followed by the parameters in parentheses.

```
CALL ICNOBPUN(parm1,parm2,parm3,. . .)
```

Refer to *NCP, SSP, and EP Resource Definition Guide* for a detailed description of these routines.

What the Utilities Can Do

NDF internal utilities provide the following general functions:

- Add or replace keywords on the current statement
- Pass statements and their keywords to NDF
- Validate character and number strings
- Print error messages to the generation report
- Pass link-edit statements to NDF
- Generate table assembly source
- Trace routines and their parameters
- Manipulate strings
- Convert number strings to numbers
- Store and retrieve data using a dynamic table storage facility
- Get the date and time of generation
- Get keyword subvalues
- Get the network address associated with a symbol
- Produce a load module separate from the NCP controller load module
- Locate insert point for user verification
- Search the SYSLIB data set for a member
- Add a comment to a generated statement group
- Get a user-coded value for any keyword on current statement
- Produce an assembly record to go into table 2 source code
- Post link-edit statements for code added to table 2.

Using the Transfer Vector Table

The transfer vector table (XVT) contains the starting addresses of the NDF internal utilities available for use by your generation application. When NDF calls your statement and keyword processing routines, it passes the table as a parameter. Table 27 on page 124 shows the XVT layout.

Table 27. Format of the Transfer Vector Table

0(0)	Number of addresses in the table
4(4)	Address of ICNCVLAB
8(8)	Address of ICNCVRNG
	•
	•
	•
204(CC)	Address of ICNLEPT2

For a complete listing of the transfer vector table, see Table 28 on page 129.

Using Variable-Length Parameter Lists

Several NDF routines accept a variable number of input parameters, called a variable-length parameter list. A variable-length parameter list allows a routine to accept a variable number of data items, such as variable portions of message strings. A variable-length parameter list may contain no input parameters, or may include any number of input parameters. The last parameter is indicated to the calling routine by setting the high-order bit of the pointer to the parameter to B'1'.

For NDF routines that use NDF string-handling control codes, NDF uses the control code in the string to determine the parameter's data type. If the actual data type does not match the control code, errors may result.

The following example shows a control string and its corresponding parameters:

```
CALL ICNRPFMT('/S/5T/S20T/F%',in_arg1,in_arg2,in_arg3)
```

The three control codes, /S, /S, and /F, require two strings and then a fullword integer. Thus, in this example, NDF would assume that `in_arg1` and `in_arg2` are string variables and that `in_arg3` is a fullword integer.

Preparing User Applications for Inclusion at Generation Time

To use your application with NDF using the standard attachment facility, you must assemble your code and link the portions to form executable load modules, called *user-written generation load modules.*

You must then code the name of each load module on the OPTIONS statement using the USERGEN keyword. This OPTIONS statement must be the first executable statement in the generation definition.

Note: No user routines can be called on the first OPTIONS statement. NDF processes all NCP keywords *after* user processing. For a given statement, user routines are called in the order they are coded on the USERGEN keyword.

This example shows the OPTIONS statement in the generation definition:

```
OPTIONS  USERGEN=(USRPROG1,USRPROG2)
```

In this example, NDF calls all of the routines in generation application USRPROG1 for a given macro before it calls any routines in generation application USRPROG2.

The name supplied with the USERGEN keyword must be the name of a load module in the STEPLIB chain.

Sample Generation Application

This sample generation application shows the mechanics of processing a statement and calling NDF internal utilities.

```
* Example of a user written generation routine
*
* Define the SKVT using the SKVT record macros defined elsewhere in
* the user written routine.
*
* Define the SKVT
* Process TST.CBGENER keyword on the GROUP statement
  TSTSKVT  CSECT
         USING *,15
         ICNVTSTR  NAME=TSTASM SKVT START RECORD FOR TSTASM
         ICNVTSTM  NAME=GROUP  SKVT STATEMENT RECORD FOR GROUP
         ICNVTKEY  ROUTINE=TSTASM,NAME=TST.CBGENER KEYWORD RECORD
         ICNVTEND            SKVT END RECORD
*
* Executable code
* Save registers, place information in save areas
* Set up addressability
* Reg 14 = return address
* Reg 13 = caller's save area address
* Reg 01 = address of input parameter pointers
  TSTASM   CSECT               Begin executable code
         USING *,15
         B PROLOG              Branch to prolog processing
         DC AL1(16)            Set routine name 5 bytes from entry point
         DC CL8'TSTASM'        Routine name for procedure trace back
  PROLOG STM 14,12,12(13)      Store regs in caller's save area
         BALR 02,0             Reg 2 = address of next instruction
         USING START,02        A set up for relocation
  START  ST 13,SAVEAREA+4      Store addr of caller's save area in ours
         LA 14,SAVEAREA        Reg 14 = address of our save area
         ST 14,8(,13)          Store addr of our save area in caller's
         MVC INPARMS(32),0(01) Store parameter pointers based on reg 1
         LR 13,14              Reg 13 = address of our save area
*
* If procedure trace bit in status word 0 then don't trace entry
*
         L 01,STATUS           Reg 1 = address of NDF status word
         TM 0(01),B'01000000'  Test procedure trace bit in status word
         BZ SKPTRC             Branch if 0 to SKPTRC
*
* Else trace procedure entry
*
         LA 15,RTNNAME         Reg 15 = address of routine name
         ST 15,OUTPARMS        Store address of first parameter
         LA 15,TRENTER         Reg 15 = address of TRENTER byte
         ST 15,OUTPARMS+4      Store address of second parameter
         L 01,XVT              Reg 1 = address of XVT
         L 15,ICNRPTRC(,01)    Reg 15 = address of ICNRPTRC routine
         LA 01,OUTPARMS        Reg 1 = address of output parameters
         BALR 14,15            Call ICNRPTRC('TSTASM%','00000000'B)
```

Figure 39 (Part 1 of 4). Sample Generation Application

```
*
* If number of subvalues ¬= one then go to THEEND
*
SKPTRC   LA 12,1               Reg 12 = 1
         L 14,SUBCNT           Reg 14 = address of subvalue count parm
         C 12,0(,14)           Compare 1 and subvalue count
         BNE THEEND            If not equal then go to the end
*
* If subvalue ¬= 'YES%' then go to THEEND
*
         L 15,SUBVALS          Reg 15 = address of subvalue parameters
         L  01,4(,15)          Reg 1 = address of first subvalue
         CLC 0(4,01),YES       Compare first subvalue with 'YES%'
         BNE THEEND            If not equal then go to the end
*
* Call ICNOBPUN('/S/10T/S%','TSTCB%','CSECT%')
* Put 'TSTCB    CSECT' in table assembly
*
         LA 15,CONTSTR         Reg 15 = address of first parameter
         ST 15,OUTPARMS        Store address of first parameter
         LA 15,LABEL           Reg 15 = address of second parameter
         ST 15,OUTPARMS+4      Store address of second parameter
         LA 15,CSCT            Reg 15 = address of third parameter
         ST 15,OUTPARMS+8      Store address of third parameter
         OI OUTPARMS+8,X'80'   Set last parameter bit
         L 01,XVT              Reg 1 = address of XVT
         L 15,ICNOBPUN(,01)    Reg 15 = address of ICNOBPUN routine
         LA 01,OUTPARMS        Reg 1 = address of output parameters
         BALR 14,15            Call ICNOBPUN routine
*
* Call ICNOBPUN('/S/10T/S%','%','DC C''TSTCB''%')
* Put '         DC C'TSTCB'' in table assembly
*
         LA 11,NULL            Load address of new second parameter
         ST 11,OUTPARMS+4      Store address of second parameter
         LA 11,INSTRCT         Load address of new third parameter
         ST 11,OUTPARMS+8      Store address of third parameter
         OI OUTPARMS+8,X'80'   Set last parameter bit
         BALR 14,15
         DS 0H
*
* If procedure trace bit in status word 0 then don't trace exit
*
THEEND   L 01,STATUS           Reg 1 = address of NDF status word
         TM 0(01),B'01000000'  Test procedure trace bit in status word
         BZ SKPTRC2            Branch if 0 to SKPTRC2
```

Figure 39 (Part 2 of 4). Sample Generation Application

```
*
* Trace procedure exit
*
         LA 15,RTNNAME          Reg 15 = address of routine name
         ST 15,OUTPARMS         Store address of first parameter
         LA 15,TREXIT           Reg 15 = address of TREXIT byte
         ST 15,OUTPARMS+4       Store address of second parameter
         L 01,XVT               Reg 1 = address of XVT
         L 15,ICNRPTRC(,01)     Reg 15 = address of ICNRPTRC routine
         LA 01,OUTPARMS         Reg 1 = address of output parameters
         BALR 14,15             Call ICNRPTRC('TSTCODE%','10000000'B)
*
* Restore registers, return to caller
*
SKPTRC2  L 13,4(,13)            Reg 13 = address of caller's save area
         LM 14,12,12(13)        Restore caller's registers
         BR 14                  Return to caller
DATA     DS 0H
         DS 0F
*
* Define storage for input parameters, output parameters, save area
*
INPARMS  DS 8F
OUTPARMS DS 8F
SAVEAREA DS 18F
*
* Define constants
*
INSTRCT  DC C'DC C''TSTCB''%'
CONTSTR  DC C'/S/10T/S%'
RTNNAME  DC C'TSTASM%'
LABEL    DC C'TSTCB%'
CSCT     DC C'CSECT%'
YES      DC C'YES%'
NULL     DC C'%'
TRENTER  DC B'00000000'
TREXIT   DC B'10000000'
*
* Define offsets for input parameters
*
STATUS   EQU INPARMS+0
STMTNAME EQU INPARMS+4
KEYNAME  EQU INPARMS+8
KEYVAL   EQU INPARMS+12
KEYLEN   EQU INPARMS+16
SUBVALS  EQU INPARMS+20
SUBCNT   EQU INPARMS+24
XVT      EQU INPARMS+28
```

Figure 39 (Part 3 of 4). Sample Generation Application

```
*
* Define offsets for XVT routines
*
XVTCOUNT EQU 0
ICNCVLAB EQU 4
ICNCVRNG EQU 8
ICNCVTOK EQU 12
ICNERPST EQU 16
ICNLEPTN EQU 20
ICNOBPUN EQU 24
ICNRPFMT EQU 28
ICNRPINF EQU 32
ICNRPPBT EQU 36
ICNRPPCH EQU 40
ICNRPPCX EQU 44
ICNRPPFX EQU 48
ICNRPPHX EQU 52
ICNRPPNM EQU 56
ICNRPTRC EQU 60
ICNSMAFT EQU 64
ICNSMALN EQU 68
ICNSMBEF EQU 72
ICNSMCAT EQU 76
ICNSMCMP EQU 80
ICNSMFMT EQU 84
ICNSMFND EQU 88
ICNSMLEN EQU 92
ICNSMPLS EQU 96
ICNSMSTD EQU 100
ICNSMSUB EQU 104
ICNSMTRM EQU 108
ICNTCBTA EQU 112
ICNTCBTC EQU 116
ICNTCDEC EQU 120
ICNTCDTC EQU 124
ICNTCDTH EQU 128
ICNTCHEX EQU 132
ICNUSKAD EQU 136
ICNUSKRP EQU 140
ICNUSNEW EQU 144
ICNUSSTA EQU 148
ICNUSSTS EQU 152
ICNUSTAD EQU 156
ICNUSTRT EQU 160
ICNUSTUP EQU 164
ICNUSDTG EQU 168
ICNUSGKI EQU 172
ICNUSRNA EQU 176
ICNLEPTS EQU 180
ICNUSSTI EQU 184
ICNUSLIB EQU 188
ICNIPGKI EQU 192
ICNUSSTC EQU 196
ICNOBPU2 EQU 200
ICNLEPT2 EQU 204
         END
```

Figure 39 (Part 4 of 4). Sample Generation Application

Transfer Vector Table

Table 28 shows the transfer vector table (XVT).

Table 28 (Page 1 of 2). Transfer Vector Table

0(0)	Number of addresses in table
4(4)	Address of ICNCVLAB
8(8)	Address of ICNCVRNG
12(C)	Address of ICNCVTOK
16(10)	Address of ICNERPST
20(14)	Address of ICNLEPTN
24(18)	Address of ICNOBPUN
28(1C)	Address of ICNRPFMT
32(20)	Address of ICNRPINF
36(24)	Address of ICNRPPBT
40(28)	Address of ICNRPPCH
44(2C)	Address of ICNRPPCX
48(30)	Address of ICNRPPFX
52(34)	Address of ICNRPPHX
56(38)	Address of ICNRPPNM
60(3C)	Address of ICNRPTRC
64(40)	Address of ICNSMAFT
68(44)	Address of ICNSMALN
72(48)	Address of ICNSMBEF
76(4C)	Address of ICNSMCAT
80(50)	Address of ICNSMCMP
84(54)	Address of ICNSMFMT
88(58)	Address of ICNSMFND
92(5C)	Address of ICNSMLEN
96(60)	Address of ICNSMPLS
100(64)	Address of ICNSMSTD
104(68)	Address of ICNSMSUB
108(6C)	Address of ICNSMTRM
112(70)	Address of ICNTCBTA
116(74)	Address of ICNTCBTC
120(78)	Address of ICNTCDEC
124(7C)	Address of ICNTCDTC
128(80)	Address of ICNTCDTH
132(84)	Address of ICNTCHEX
136(88)	Address of ICNUSKAD
140(8C)	Address of ICNUSKRP

Table 28 (Page 2 of 2). Transfer Vector Table

144(90)	Address of ICNUSNEW
148(94)	Address of ICNUSSTA
152(98)	Address of ICNUSSTS
156(9C)	Address of ICNUSTAD
160(A0)	Address of ICNUSTRT
164(A4)	Address of ICNUSTUP
168(A8)	Address of ICNUSDTG
172(AC)	Address of ICNUSGKI
176(B0)	Address of ICNUSRNA
180(B4)	Address of ICNLEPTS
184(B8)	Address of ICNUSSTI
188(BC)	Address of ICNUSLIB
192(C0)	Address of ICNIPGKI
196(C4)	Address of ICNUSSTC
200(C8)	Address of ICNOBPU2
204(CC)	Address of ICNLEPT2

Glossary, Bibliography, and Index

Glossary

This glossary includes terms and definitions from:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- The ANSI/EIA Standard—440-A, *Fiber Optic Terminology*. Copies may be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue, N.W., Washington, DC 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.
- The Network Working Group Request for Comments: 1208.

The following cross-references are used in this glossary:

Contrast with: This refers to a term that has an opposed or substantively different meaning.

Synonym for: This indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the glossary.

Synonymous with: This is a backward reference from a defined term to all other terms that have the same meaning.

See: This refers the reader to multiple-word terms that have the same last word.

See also: This refers the reader to terms that have a related, but not synonymous, meaning.

Deprecated term for: This indicates that the term should not be used. It refers to a preferred term, which is defined in its proper place in the glossary.

A

abend. (1) Abnormal end of task. (2) Synonym for *abnormal termination.*

abnormal end. Synonym for *abnormal termination.*

abnormal end of task (abend). Termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task is executing.

abnormal termination. (1) The cessation of processing prior to planned termination. (T) (2) A system failure or operator action that causes a job to end unsuccessfully. (3) Synonymous with *abend* and *abnormal end.*

ACB. (1) In VTAM, access method control block. (2) In NCP, adapter control block. (3) Application control block.

ACB name. (1) The name of an ACB macroinstruction. (2) A name specified either on the VTAM APPL definition statement or on the VTAM application program's ACB macroinstruction. Contrast with *network name.*

ACCESS. In the Simple Network Management Protocol (SNMP), the clause in a Management Information Base (MIB) module that defines the minimum level of support that a managed node provides for an object.

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

ACF/TAP. Advanced Communications Function/Trace Analysis Program. Synonymous with *TAP.*

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

acknowledgment. (1) The transmission, by a receiver, of acknowledge characters as an affirmative response to a sender. (T) (2) An indication that an item sent was received.

action. (1) In the AIX operating system, a defined task that an application performs. An action modifies the properties of an object or manipulates the object in some way. (2) An operation on a managed object, the semantics of which are defined as part of the managed object class definition.

ACTLINK. Activate link.

ACTLU. Activate logical unit. In SNA, a command used to start a session on a logical unit.

ACTPU. Activate physical unit. In SNA, a command used to start a session on a physical unit.

ACU. Automatic calling unit.

adapter. A part that electrically or physically connects a device to a computer or to another device.

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

adaptive pacing. Synonym for *adaptive session-level pacing* and *virtual route pacing*.

adaptive session-level pacing. A form of session-level pacing in which session components exchange pacing windows that may vary in size during the course of a session. This allows transmission within a network to adapt dynamically to variations in availability and demand of buffers on a session-by-session basis. Session-level pacing occurs within independent stages along the session path according to local congestion at the intermediate and endpoint nodes. Synonymous with *adaptive pacing* and *adaptive session pacing*. See *pacing*, *session-level pacing*, and *virtual route pacing*.

adaptive session pacing. Synonym for *adaptive session-level pacing*.

address space manager (ASM). A component in an APPN or LEN node that assigns and frees session addresses.

adjacent link station (ALS). (1) In SNA, a link station directly connected to a given node by a link connection over which network traffic can be carried.

Note: Several secondary link stations that share a link connection do not exchange data with each other and therefore are not adjacent to each other. (2) With respect to a specific node, a link station partner in an adjacent node.

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

Advanced Communications Function/Trace Analysis Program (ACF/TAP). An SSP program service aid that assists in analyzing trace data produced by VTAM, TCAM, and NCP and provides network data traffic and network error reports. Synonymous with *Trace Analysis Program (TAP)*.

Advanced Peer-to-Peer Networking (APPN). An extension to SNA featuring (a) greater distributed network control that avoids critical hierarchical dependencies, thereby isolating the effects of single points of failure; (b) dynamic exchange of network topology information to foster ease of connection, reconfiguration, and adaptive route selection; (c) dynamic definition of network resources; and (d) automated resource registration and directory lookup. APPN extends the LU 6.2 peer orientation for end-user services to network control and supports multiple LU types, including LU 2, LU 3, and LU 6.2.

Advanced Peer-to-Peer Networking (APPN) end node. A node that provides a broad range of end-user services and supports sessions between its local control point (CP) and the CP in an adjacent network node. It uses these sessions to dynamically register its resources with the adjacent CP (its network node server), to send and receive directory search requests, and to obtain management services. An APPN end node can also attach to a subarea network as a peripheral node or to other end nodes.

Advanced Peer-to-Peer Networking (APPN) network. A collection of interconnected network nodes and their client end nodes.

Advanced Peer-to-Peer Networking (APPN) network node. A node that offers a broad range of end-user services and that can provide the following:

- Distributed directory services, including registration of its domain resources to a central directory server
- Topology database exchanges with other APPN network nodes, enabling network nodes throughout the network to select optimal routes for LU-LU sessions based on requested classes of service
- Session services for its local LUs and client end nodes
- Intermediate routing services within an APPN network

Advanced Peer-to-Peer Networking (APPN) node. An APPN network node or an APPN end node.

alert. (1) A message sent to a management services focal point in a network to identify a problem or an impending problem. (2) In the NetView and NETCENTER programs, a high priority event that warrants immediate attention.

alias address. An address used by a gateway NCP and a gateway system services control point (SSCP) in

one network to represent a logical unit (LU) or SSCP in another network.

allocate. A logical unit (LU) 6.2 application program interface (API) verb used to assign a session to a conversation for the conversation's use. Contrast with *deallocate.*

ANA. Assign network address.

AND operation. Synonym for *conjunction.*

appendage. An application program routine provided to assist in handling a specific occurrence.

application control block (ACB). The control blocks created from the output of DBDGEN and PSBGEN and placed in the ACB library for use during online and DBB region type execution of IMS/VS.

apply. An SMP process that moves distributed code and MVS-type programs to the system libraries.

APPN. Advanced Peer-to-Peer Networking.

area. In Internet and DECnet routing protocols, a subset of a network or gateway grouped together by definition of the network administrator. Each area is self-contained; knowledge of an area's topology remains hidden from other areas.

argument. (1) An independent variable. (I) (A) (2) Any value of an independent variable; for example, a search key; a number identifying the location of an item in a table. (I) (A) (3) A parameter passed between a calling program and a called program.

ASM. Address space manager.

automatic calling unit (ACU). A dialing device that permits a computer to automatically dial calls over a network.

autotask. An unattended NetView operator station task that does not require a terminal or a logged-on user. Autotasks can run independently of VTAM and are typically used for automated console operations. Contrast with *logged-on operator.*

B

basic encoding rules (BER). The rules specified in ISO 8825 for encoding data units described in abstract syntax notation 1 (ASN.1). The rules specify the encoding technique, not the abstract syntax.

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

BER. (1) Box event record. (2) Box error record. (3) Basic encoding rules.

binary synchronous communication (BSC). A form of telecommunication line control that uses a standard set of transmission control characters and control character sequences, for binary synchronous transmission of binary-coded data between stations. Contrast with *Synchronous Data Link Control (SDLC).*

binary synchronous transmission. Data transmission in which synchronization of characters is controlled by timing signals generated at the sending and receiving stations. See also *start-stop transmission* and *Synchronous Data Link Control (SDLC).*

BIND. In SNA, a request to activate a session between two logical units (LUs). See also *session activation request.* Contrast with *UNBIND.*

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

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

Note: When blocking is not done, a BTU consists of one PIU.

Boolean. (1) Pertaining to the processes used in the algebra formulated by George Boole. (A) (2) A value of 0 or 1 represented internally in binary notation.

boundary function. (1) In SNA, a capability of a subarea node to provide protocol support for attached peripheral nodes, such as: (a) interconnecting subarea path control and peripheral path control elements, (b) performing session sequence numbering for low-function peripheral nodes, and (c) providing session-level pacing support. (2) In SNA, the component that provides these capabilities.

boundary node (BN). In SNA, a subarea node with boundary function.

Note: A subarea node may be a boundary node, an intermediate routing node, both, or neither, depending on how it is used in the network.

bracket. In SNA, one or more chains of request units and their responses that are exchanged between two session partners and that represent a transaction between them. A bracket must be completed before another bracket can be started. Examples of brackets

are database inquiries/replies, update transactions, and remote job entry output sequences to workstations.

bracket protocol. In SNA, a data flow control protocol in which exchanges between two session partners are achieved through the use of brackets, with one partner designated at session activation as the first speaker and the other as the bidder. The bracket protocol involves bracket initiation and termination rules.

bridge. (1) A functional unit that interconnects two local area networks that use the same logical link control protocol but may use different medium access control protocols. (T) (2) A functional unit that interconnects multiple LANs (locally or remotely) that use the same logical link control protocol but that can use different medium access control protocols. A bridge forwards a frame to another bridge based on the medium access control (MAC) address. (3) In the connection of local loops, channels, or rings, the equipment and techniques used to match circuits and to facilitate accurate data transmission. (4) See also *gateway.*

BSC. Binary synchronous communication.

BTU. Basic transmission unit.

buffer. (1) A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another. (A) (2) To allocate and schedule the use of buffers. (A) (3) A portion of storage used to hold input or output data temporarily.

bus. (1) A facility for transferring data between several devices located between two end points, only one device being able to transmit at a given moment. (T) (2) A computer configuration in which processors are interconnected in series.

C

call. (1) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (I) (A) (2) In data communication, the actions necessary to make a connection between two stations on a switched line. (3) In communications, a conversation between two users. (4) To transfer control to a procedure, program, routine, or subroutine. (5) To attempt to contact a user, regardless of whether the attempt is successful.

calling. (1) The process of transmitting selection signals in order to establish a connection between data stations. (I) (A) (2) In X.25 communications, pertaining to the location or user that makes a call.

CCU. Central control unit.

CDS. (1) Control data set. (2) Configuration data set. (3) Central directory server.

central directory server (CDS). A network node that provides a repository for information on network resource locations; it also reduces the number of network searches by providing a focal point for queries and broadcast searches and by caching the results of network searches to avoid later broadcasts for the same information.

chain. (1) A group of logically linked user data records processed by LU 6.2. (2) A group of request units delimited by begin-chain and end-chain. Responses are always single-unit chains. See *RU chain.*

channel. (1) A path along which signals can be sent, for example, data channel, output channel. (A) (2) A functional unit, controlled by the processor, that handles the transfer of data between processor storage and local peripheral equipment. See *input/output channel.*

channel adapter. A communication controller hardware unit that is used to attach the communication controller to a host channel.

circuit. (1) One or more conductors through which an electric current can flow. See *physical circuit* and *virtual circuit.* (2) A logic device.

circuit switching. (1) A process that, on demand, connects two or more data terminal equipment (DTEs) and permits the exclusive use of a data circuit between them until the connection is released. (I) (A) (2) Synonymous with *line switching.* (3) See also *message switching* and *packet switching.*

cleanup. In SNA products, a network services request, sent by a system services control point (SSCP) to a logical unit (LU), that causes a particular LU-LU session with that LU to be ended immediately without requiring the participation of either the other LU or its SSCP.

cluster. (1) A station that consists of a control unit (a cluster controller) and the terminals attached to it. (2) A group of APPN nodes that have the same network ID and the same topology database. A cluster is a subset of a NETID subnetwork.

CMS. Conversational monitor system.

communication controller. A type of communication control unit whose operations are controlled by one or more programs stored and executed in the unit. It manages the details of line control and the routing of data through a network.

communication scanner processor (CSP). A processor in the 3725 Communication Controller that

contains a microprocessor with control code. The code controls transmission of data over links attached to the CSP.

configuration report program (CRP). An SSP utility program that creates a configuration report listing network resources and resource attributes for networks with NCP, EP, PEP, or VTAM.

conjunction. The Boolean operation whose result has the Boolean value 1 if and only if each operand has the Bolean value 1. (I) (A) Synonymous with *AND operation.*

connection. (1) In data communication, an association established between functional units for conveying information. (I) (A) (2) In Open Systems Interconnection architecture, an association established by a given layer between two or more entities of the next higher layer for the purpose of data transfer. (T) (3) In VTAM, synonym for *physical connection.* (4) In SNA, the network path that links together two logical units (LUs) in different nodes to enable them to establish communications. (5) In X.25 communication, a virtual circuit between two data terminal equipments (DTEs). A switched virtual circuit (SVC) connection lasts for the duration of a call; a permanent virtual circuit (PVC) is a permanent connection between the DTEs. (6) In TCP/IP, the path between two protocol applications that provides reliable data stream delivery service. In Internet, a connection extends from a TCP application on one system to a TCP application on another system.

connection point manager. In SNA, a component of the transmission control layer that: (1) performs session-level pacing of normal-flow requests, (2) checks sequence numbers of received request units, (3) verifies that request units do not exceed the maximum permissible size, (4) routes incoming request units to their destinations in the half-session, and (5) enciphers and deciphers FMD request units when cryptography is selected. The connection point manager coordinates the normal and expedited flows for one half-session.

Note: The sending connection point manager in a half-session builds the request/response header (RH) for outgoing request/response units (RUs), and the receiving connection point manager interprets the request/response headers that precede incoming request/response units.

constraints. In NETDA/2, the set of essential requirements specified with the node, connection, or application definitions. A change in a constraint value changes the input to the network design. Contrast with *parameters.*

contention. In a session, a situation in which both NAUs attempt to initiate the same action at the same time, such as when both attempt to send data in a half-duplex protocol (half-duplex contention), or both attempt to start a bracket (bracket contention). At session initiation, one NAU is defined to be the contention winner; its action will take precedence when contention occurs. The contention loser must get explicit or implicit permission from the contention winner to begin its action.

control block. (1) A storage area used by a computer program to hold control information. (I) (2) In the IBM Token-Ring Network, a specifically formatted block of information provided from the application program to the Adapter Support Interface to request an operation.

control data set (CDS). In NPM, an SMP data set used in the NPM installation process.

control point (CP). (1) A component of an APPN or LEN node that manages the resources of that node. In an APPN node, the CP is capable of engaging in CP-CP sessions with other APPN nodes. In an APPN network node, the CP also provides services to adjacent end nodes in the APPN network. (2) A component of a node that manages resources of that node and optionally provides services to other nodes in the network. Examples are a system services control point (SSCP) in a type 5 subarea node, a network node control point (NNCP) in an APPN network node, and an end node control point (ENCP) in an APPN or LEN end node. An SSCP and an NNCP can provide services to other nodes.

control point management services (CPMS). A component of a control point, consisting of management services function sets, that provides facilities to assist in performing problem management, performance and accounting management, change management, and configuration management. Capabilities provided by the CPMS include sending requests to physical unit management services (PUMS) to test system resources, collecting statistical information (for example, error and performance data) from PUMS about the system resources, and analyzing and presenting test results and statistical information collected about the system resources. Analysis and presentation responsibilities for problem determination and performance monitoring can be distributed among multiple CPMSs.

control program. (1) A computer program designed to schedule and to supervise the execution of programs of a computer system. (I) (A) (2) The part of the AIX Base Operating System that determines the order in which basic functions should be performed. (3) See *VM/370 control program (CP).*

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

control vector. One of a general class of RU substructures that has variable length, is carried within some enclosing structure, and has a one-byte key used as an identifier.

conversational monitor system (CMS). A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities, and operates only under control of the VM/370 control program.

CP. (1) VM/370 control program. (2) Control point.

CPMS. Control point management services.

cross-domain. In SNA, pertaining to control or resources involving more than one domain.

cross-network session. An LU-LU or SSCP-SSCP session whose path traverses more than one SNA network.

CRP. Configuration report program.

CWALL. An NCP threshold of buffer availability, below which the NCP will accept only high-priority path information units (PIUs).

D

DAF. Destination address field.

DAF'. Destination address field prime.

data channel. Synonym for *input/output channel.*

data circuit. (1) A pair of associated transmit and receive channels that provide a means of two-way data communication. (I) (2) In SNA, synonym for *link connection.* (3) See also *physical circuit* and *virtual circuit.*

Notes:

1. Between data switching exchanges, the data circuit may include data circuit-terminating equipment (DCE), depending on the type of interface used at the data switching exchange.
2. Between a data station and a data switching exchange or data concentrator, the data circuit includes the data circuit-terminating equipment at the data station end, and may include equipment similar to a DCE at the data switching exchange or data concentrator location.

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

data link control (DLC). A set of rules used by nodes on a data link (such as an SDLC link or a token ring) to accomplish an orderly exchange of information.

data link level. (1) In the hierarchical structure of a data station, the conceptual level of control or processing logic between high level logic and the data link that maintains control of the data link. The data link level performs such functions as inserting transmit bits and deleting receive bits; interpreting address and control fields; generating, transmitting, and interpreting commands and responses; and computing and interpreting frame check sequences. See also *higher level, packet level,* and *physical level.* (2) In X.25 communications, synonym for *frame level.*

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

data stream. (1) All information (data and control commands) sent over a data link usually in a single read or write operation. (2) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

DC. Data chaining.

ddname. Data definition name.

deallocate. A logical unit (LU) 6.2 application program interface (API) verb that terminates a conversation, thereby freeing the session for a future conversation. Contrast with *allocate.*

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

definition statement. In NCP, a type of instruction that defines a resource to the NCP. See Figure 40. See also *macroinstruction.*

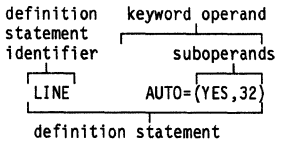


Figure 40. Example of an NCP Definition Statement

dependent LU. See *SSCP-dependent LU.*

destination address field (DAF). In SNA, a field in a FID0 or FID1 transmission header that contains the network address of the destination.

dial-out. Pertaining to the direction in which a switched connection is requested by a host or an NCP.

direct memory access (DMA). The system facility that allows a device on the Micro Channel bus to get direct access to the system or bus memory without the intervention of the system processor.

directory. (1) A table of identifiers and references to the corresponding items of data. (I) (A) (2) A database in an APPN node that lists names of resources (in particular, logical units) and records the CP name of the node where each resource is located. See *distributed directory database* and *local directory database.* (3) In VM/ESA, a control program (CP) disk file that defines each virtual machine's normal configuration: the user ID, password, normal and maximum allowable virtual storage, CP command privilege classes allowed, dispatching priority, logical editing symbols to be used, account number, and CP options desired.

directory service (DS). An application service element that translates the symbolic names used by application processes into the complete network addresses used in an OSI environment. (T)

directory services (DS). A control point component of an APPN node that maintains knowledge of the location of network resources.

distributed directory database. The complete listing of all the resources in the network as maintained in the individual directories scattered throughout an APPN network. Each node has a piece of the complete directory, but it is not necessary for any one node to have the entire list. Entries are created, modified, and deleted through system definition, operator action, automatic registration, and ongoing network search procedures. Synonymous with *distributed network directory* and *network directory database.*

distributed network directory. Synonym for *distributed directory database.*

DMA. Direct memory access.

domain. (1) That part of a computer network in which the data processing resources are under common control. (T) (2) In SNA, see *end node domain, network node domain,* and *system services control point domain.* (3) In the Internet, a part of a naming hierarchy in which the domain name consists of a sequence of names (labels) separated by periods (dots). (4) In Open Systems Interconnection (OSI), a part of a distributed system or a set of managed objects to which a common policy applies.

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

DSECT. Dummy control section.

duplex. Pertaining to communication in which data can be sent and received at the same time. Synonymous with *full duplex.* Contrast with *half duplex.*

E

ECB. Event control block.

element. (1) A field in the network address. (2) In SNA, the particular resource within a subarea that is identified by an element address. See also *subarea.*

element address. In SNA, a value in the element address field of the network address identifying a specific resource within a subarea. See *subarea address.*

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

Emulation Program (EP). An IBM control program that allows a channel-attached 3705 or 3725 communication controller to emulate the functions of an IBM 2701 Data Adapter Unit, an IBM 2702 Transmission Control, or an IBM 2703 Transmission Control. See also *network control program.*

end node domain. An end node control point, its attached links, and its local LUs.

entry point (EP). In SNA, a type 2.0, type 2.1, type 4, or type 5 node that provides distributed network management support. It sends network management data about itself and the resources it controls to a focal point for centralized processing, and it receives and executes focal-point initiated commands to manage and control its resources.

EP. (1) Emulation Program. (2) Entry point.

ER. (1) Explicit route. (2) Exception response.

ESC. Execution sequence control.

Ethernet. A 10-Mbps baseband local area network that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and transmission. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD).

event control block (ECB). A control block used to represent the status of an event.

exception. An abnormal condition such as an I/O error encountered in processing a data set or a file.

exception response (ER). In SNA, a protocol requested in the form-of-response-requested field of a request header that directs the receiver to return a response only if the request is unacceptable as received or cannot be processed; that is, a negative response, but not a positive response, can be returned. Contrast with *definite response* and *no response.*

exchange identification (XID). A specific type of basic link unit that is used to convey node and link characteristics between adjacent nodes. XIDs are exchanged between link stations before and during link activation to establish and negotiate link and node characteristics, and after link activation to communicate changes in these characteristics.

exit. (1) To execute an instruction within a portion of a computer program in order to terminate the execution of that portion. Such portions of computer programs include loops, subroutines, modules, and so on. (T) (2) See *installation exit* and *user exit.*

explicit route (ER). In SNA, a series of one or more transmission groups that connect two subarea nodes. An explicit route is identified by an origin subarea address, a destination subarea address, an explicit route number, and a reverse explicit route number. Contrast with *virtual route (VR).*

extended architecture (XA). An extension to System/370 architecture that takes advantage of continuing high performance enhancements to computer system hardware.

extended binary-coded decimal interchange code (EBCDIC). A coded character set of 256 8-bit characters.

F

fallback. In an IBM 3745 Communication Controller with twin central control units (CCUs) in standby or backup mode, the process by which buses are switched from the failing CCU to the active CCU (backup mode) or the idle CCU (standby mode) to recover the path of communication flow in the failed CCU. See also *switchback.*.

FASTRUN. One of several options available with the NCP/EP definition facility (NDF) that indicates that only the syntax is to be checked in generation definition statements.

FID. Format identification.

FIFO. First-in-first-out. (A)

flow. In NETDA/2, the amount of traffic that can pass through a node, connection, or route in both directions during a given period of time.

flow control. In SNA, the process of managing the rate at which data traffic passes between components of the network. The purpose of flow control is to optimize the rate of flow of message units with minimum congestion in the network; that is, to neither overflow the buffers at the receiver or at intermediate routing nodes, nor leave the receiver waiting for more message units. See also *adaptive session-level pacing*, *pacing*, and *session-level pacing.*

format identification (FID) field. In SNA, a field in each transmission header (TH) that indicates the format of the TH; that is, the presence or absence of certain fields. TH formats differ in accordance with the types of nodes between which they pass. Following are the six FID types:

FID0, used for traffic involving non-SNA devices between adjacent subarea nodes when either or both nodes do not support explicit route and virtual route protocols

FID1, used for traffic involving SNA devices between adjacent subarea nodes when either or both nodes do not support explicit route and virtual route protocols

FID2, used for traffic between a subarea node and an adjacent type 2 peripheral node

FID3, used for traffic between a subarea node and an adjacent type 1 peripheral node

FID4, used for traffic between adjacent subarea nodes when both nodes support explicit route and virtual route protocols

FIDF, used for certain commands (for example, for transmission group control) sent between adjacent subarea nodes when both nodes support explicit route and virtual route protocols.

frame. (1) In Open Systems Interconnection architecture, a data structure pertaining to a particular area of knowledge and consisting of slots that can accept the values of specific attributes and from which inferences can be drawn by appropriate procedural attachments. (T) (2) The unit of transmission in some local area networks, including the IBM Token-Ring Network. It includes delimiters, control characters, information, and checking characters. (3) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures.

frame level. See *link level.*

full duplex (FDX). Synonym for *duplex.*

function management data (FMD). An RU category used for end-user data exchanged between logical units (LUs) and for requests and responses exchanged between network services components of LUs, PUs, and control points.

G

gateway. (1) A functional unit that interconnects two computer networks with different network architectures. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures. (T) (2) In the AIX operating system, an entity that operates above the link layer and translates, when required, the interface and protocol used by one network into those used by another distinct network. (3) In TCP/IP, a device used to connect two systems that use either the same or different communications protocols. (4) The combination of machines and programs that provide address translation, name translation, and system services control point (SSCP) rerouting between independent SNA networks to allow those networks to communicate. A gateway consists of one gateway NCP and at least one gateway VTAM. (5) In the IBM Token-Ring Network, a device and its associated software that connect a local area network to another local area network or a host that uses different logical link protocols.

gateway NCP. An NCP that performs address translation to allow cross-network session traffic. The gateway NCP connects two or more independent SNA networks. Synonymous with *gateway node.*

gateway node. Synonym for *gateway NCP.*

generic alert. A product-independent method of encoding alert data by means of both (a) code points indexing short units of stored text and (b) textual data.

generic unbind. Synonym for *session deactivation request.*

H

half-duplex (HD, HDX). In data communication, pertaining to transmission in only one direction at a time. Contrast with *duplex.* See also *half-duplex operation* and *half-duplex transmission.*

half-duplex operation. A mode of operation of a data link in which data can be transmitted in both directions, one way at a time. (T)

half-duplex transmission. Data transmission in either direction, one direction at a time. (I) (A)

header. (1) System-defined control information that precedes user data. (2) The portion of a message that contains control information for the message such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

host ID. In TCP/IP, that part of the Internet address that defines the host on the network. The length of the host ID depends on the type of network or network class (A, B, or C).

host processor. (1) A processor that controls all or part of a user application network. (T) (2) In a network, the processing unit in which the data communication access method resides.

hypertext link. A pointer from a location in an online book to another location in the same book or another book. When selected, a hypertext link enables you to move quickly to the new location containing related information. BookManager associates terms with related information such as the glossary, a message or code, an index entry, or a language element reference. Cross-references indicated by markup are automatically linked to the referenced location.

I

I/O. Input/output.

ID. (1) Identifier. (2) Identification.

information (I) format. A format used for information transfer.

information (I) frame. A frame in I format used for numbered information transfer.

initial program load (IPL). (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage at the beginning of a work day or after a system malfunction. (3) The process of loading system programs and preparing a system to run jobs. (4) Synonymous with *system restart* and *system startup.*

INITIATE. A network services request sent from a logical unit (LU) to a system services control point (SSCP) requesting that an LU-LU session be established.

input/output channel. (1) In a data processing system, a functional unit that handles transfer of data between internal and peripheral equipment. (I) (A) (2) In a computing system, a functional unit, controlled by a processor, that handles transfer of data between

processor storage and local peripheral devices. Synonymous with *data channel*. See *channel*. See also *link*.

insert. In LANs, to make an attaching device an active part of the LAN.

installation exit. The means specifically described in an IBM software product's documentation by which an IBM software product may be modified by a customer's system programmers to change or extend the functions of the IBM software product. Such modifications consist of exit routines written to replace one or more existing modules of an IBM software product, or to add one or more modules or subroutines to an IBM software product, for the purpose of modifying or extending the functions of the IBM software product. Synonymous with *installation-wide exit*. See *user exit*.

installation exit routine. A routine written by a user to take control at an installation exit of an IBM software product.

installation-wide exit. Synonym for *installation exit*.

interconnection. See *SNA network interconnection (SNI)*.

intermediate node. A node that is at the end of more than one branch. (T)

Internet Protocol (IP). A connectionless protocol that routes data through a network or interconnected networks. IP acts as an intermediary between the higher protocol layers and the physical network. However, this protocol does not provide error recovery and flow control and does not guarantee the reliability of the physical network.

IP. Internet Protocol.

IPL. (1) Initial program loader. (A) (2) Initial program load.

J

JCL. Job control language.

job control language (JCL). A control language used to identify a job to an operating system and to describe the job's requirements.

K

keyword. (1) In programming languages, a lexical unit that, in certain contexts, characterizes some language construct; for example, in some contexts, IF characterizes an if-statement. A keyword normally has the form of an identifier. (I) (2) One of the predefined words of an artificial language. (A) (3) A significant and informative word in a title or document that describes the content of that document. (4) A name or symbol that identifies a parameter. (5) The part of a command operand that consists of a specific character string (such as DSNAME=). See also *definition statement* and *keyword operand*. Contrast with *positional operand*.

keyword operand. An operand that consists of a keyword followed by one or more values (such as DSNAME=HELLO). See also *definition statement*. Contrast with *positional operand*.

keyword parameter. A parameter that consists of a keyword followed by one or more values.

L

LAN. Local area network.

last-in-chain (LIC). A request unit (RU) whose request header (RH) end chain indicator is on and whose RH begin chain indicator is off. See also *RU chain*.

LCB. Local block common.

LCD. Line control definer.

LIC. (1) Last-in-chain. (2) In NCP, line interface coupler.

line. (1) The portion of a data circuit external to data circuit-terminating equipment (DCE), that connects the DCE to a data switching exchange (DSE), that connects a DCE to one or more other DCEs, or that connects a DSE to another DSE. (I) (2) Synonymous with *channel* and *circuit*.

line control discipline. Synonym for *link protocol*.

line discipline. Synonym for *link protocol*.

line group. One or more telecommunication lines of the same type that can be activated and deactivated as a unit.

line switching. Synonym for *circuit switching*.

link. (1) The combination of the link connection (the transmission medium) and two link stations, one at each end of the link connection. A link connection can be shared among multiple links in a multipoint or token-ring configuration. (2) To interconnect items of data or portions of one or more computer programs: for example, the linking of object programs by a linkage editor, linking of data items by pointers. (T)

link connection. The physical equipment providing two-way communication between one link station and

one or more other link stations; for example, a telecommunication line and data circuit-terminating equipment (DCE). Synonymous with *data circuit*.

link connection segment. A portion of the configuration that is located between two resources listed consecutively in the service point command service (SPCS) query link configuration request list.

link level. A part of Recommendation X.25 that defines the link protocol used to get data into and out of the network across the full-duplex link connecting the subscriber's machine to the network node. LAP and LAPB are the link access protocols recommended by the CCITT. See *data link level*.

Link Problem Determination Aid (LPDA). A series of procedures that are used to test the status of and to control DCEs, the communication line, and the remote device interface. These procedures, or a subset of them, are implemented by host programs (such as the NetView program and VTAM), communication controller programs (such as NCP), and IBM LPDA DCEs. See also *LPDA-1* and *LPDA-2*.

link protocol. (1) The rules for sending and receiving data at the link level. (2) See *protocol*. (3) See also *link level*.

link station. (1) The hardware and software components within a node representing a connection to an adjacent node over a specific link. For example, if node A is the primary end of a multipoint line that connects to three adjacent nodes, node A will have three link stations representing the connections to the adjacent nodes. See also *adjacent link station*. (2) In VTAM, a named resource within an APPN or a subarea node that represents the connection to another APPN or subarea node that is attached by an APPN or a subarea link. In the resource hierarchy in a subarea network, the link station is subordinate to the subarea link.

link status (LS). Information maintained by local and remote modems.

load module. All or part of a computer program in a form suitable for loading into main storage for execution. A load module is usually the output of a linkage editor. (T)

local area network (LAN). (1) A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. (T) (2) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network. See also *Ethernet* and *token ring*. (3) Contrast with *metropolitan area network (MAN)* and *wide area network (WAN)*.

local directory database. That set of resources (LUs) in the network known at a particular node. The resources included are all those in the node's domain as well as any cache entries.

local session identification (LSID). In SNA, a field in an FID3 (format identification type 3) transmission header that contains an indication of the type of session (SSCP-PU, SSCP-LU, or LU-LU) and the local address of the peripheral logical unit (LU) or physical unit (PU).

Locate. Synonym for *Locate/CD-Initiate*.

Locate/CD-Initiate. An abbreviated term for a message exchanged between APPN nodes that contains one of the following sets of general data stream (GDS) variables:

- A Locate, a Find Resource, and a Cross-Domain Initiate GDS variable used for a network search request
- A Locate, a Found Resource, and a Cross-Domain Initiate GDS variable used for a search reply when a network resource has been located

These message structures correspond to the CP components that perform the search of the distributed network directory and establish the session. The Locate GDS variable contains information used to control the delivery of the search messages in the network. The Find and Found GDS variables contain information used in the directories: origin cache data (control point information) and search arguments (destination LU name), and located resource information, respectively. The Cross-Domain Initiate GDS variable contains endpoint TG vector information to be used in selecting the route for the session. The length of the Locate/CD-Initiate message is limited to 1024 bytes.

logged-on operator. A NetView operator station task that requires a terminal and a logged-on user. Contrast with *autotask*.

logical unit (LU). A type of network accessible unit that enables end users to gain access to network resources and communicate with each other.

logical unit (LU) 6.2. A type of logical unit that supports general communication between programs in a distributed processing environment. LU 6.2 is characterized by (a) a peer relationship between session partners, (b) efficient utilization of a session for multiple transactions, (c) comprehensive end-to-end error processing, and (d) a generic application program interface (API) consisting of structured verbs that are mapped into a product implementation.

LPDA. Link Problem Determination Aid.

LPDA-1. The first version of the LPDA command set. LPDA-1 is not compatible with LPDA-2. See also *Link Problem Determination Aid (LPDA)* and *LPDA-2*.

LPDA-2. The second version of the LPDA command set. LPDA-2 provides all of the functions of LPDA-1; it also supports commands such as the following:

- DCE configuration
- Dial
- Set transmit speed
- Commands to operate a contact that can control external devices.

See also *Link Problem Determination Aid (LPDA)* and *LPDA-1*.

LSID. Local session identification.

LU-LU session. A logical connection between two logical units (LUs) in an SNA network that typically provides communication between two end users.

LUCB. Logical unit control block.

M

macroinstruction. (1) An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language and that may also specify values for parameters in the replaced instructions. (T) (2) In assembler programming, an assembler language statement that causes the assembler to process a predefined set of statements called a macro definition. The statements normally produced from the macro definition replace the macroinstruction in the program. See also *definition statement*.

maintenance and operator subsystem (MOSS). A subsystem of an IBM communication controller, such as the 3725 or the 3720, that contains a processor and operates independently of the rest of the controller. It loads and supervises the controller, runs problem determination procedures, and assists in maintaining both hardware and software.

MAN. Metropolitan area network.

management services (MS). (1) One of the types of network services in control points (CPs) and physical units (PUs). Management services are the services provided to assist in the management of SNA networks, such as problem management, performance and accounting management, configuration management, and change management. (2) Services that assist in the management of systems and networks in areas such as problem management, performance management, business management, operations management, configuration management, and change management.

mask. (1) A pattern of characters used to control retention or elimination of portions of another pattern of characters. (I) (A) (2) To use a pattern of characters to control retention or elimination of portions of another pattern of characters. (I) (A)

message. (1) An assembly of characters and sometimes control codes that is transferred as an entity from an originator to one or more recipients. A message consists of two parts: envelope and content. (T) (2) In VTAM, the amount of function management data (FMD) transferred to VTAM by the application program with one SEND request.

message switching. The process of receiving a message, storing it, and forwarding it to its destination unaltered. (T)

method. In the NetView program, the code that runs within the Resource Object Data Manager (RODM) address space. Methods are used to implement behavior specified by an operation.

metric. In Internet communications, a value, associated with a route, which is used to discriminate between multiple exit or entry points to the same autonomous system. The route with the lowest metric is preferred.

metropolitan area network (MAN). A network formed by the interconnection of two or more networks which may operate at higher speed than those networks, may cross administrative boundaries, and may use multiple access methods. (T) Contrast with *local area network (LAN)* and *wide area network (WAN)*.

microcode. (1) One or more microinstructions. (2) A code, representing the instructions of an instruction set, that is implemented in a part of storage that is not program-addressable. (3) To design, write, and test one or more microinstructions.

mixed-media multilink transmission group (MMMLTG). See *transmission group (TG)*.

MLTG. Multilink transmission group.

MMMLTG. Mixed-media multilink transmission group.

mode. See *mode name*.

mode name. The name used by the initiator of a session to designate the characteristics desired for the session, such as traffic pacing values, message-length limits, sync point and cryptography options, and the class of service within the transport network.

modem (modulator/demodulator). (1) A functional unit that modulates and demodulates signals. One of the functions of a modem is to enable digital data to be transmitted over analog transmission facilities. (T) (A) (2) A device that converts digital data from a computer to an analog signal that can be transmitted on a telecommunication line, and converts the analog signal received to data for the computer.

MOSS. Maintenance and operator subsystem.

MS. Management services.

multilink transmission group (MLTG). See *transmission group (TG).*

Multiple Virtual Storage (MVS). See *MVS.*

MVS. Multiple Virtual Storage. Implies MVS/370, the MVS/XA product, and the MVS/ESA product.

MVS/ESA product. Multiple Virtual Storage/Enterprise Systems Architecture.

MVS/XA product. Multiple Virtual Storage/Extended Architecture product, consisting of MVS/System Product Version 2 and the MVS/XA Data Facility Product, operating on a System/370 processor in the System/370 extended architecture mode. MVS/XA allows virtual storage addressing to 2 gigabytes. See also *MVS.*

N

NAU. (1) Network accessible unit. (2) Network addressable unit.

NCP. Network Control Program.

NCP/EP definition facility (NDF). A program that is part of System Support Programs (SSP) and that is used to generate a load module for a partitioned emulation program (PEP), a Network Control Program (NCP), or an Emulation Program (EP).

NCP/Token-Ring interconnection (NTRI). An NCP function that allows a communication controller to attach to the IBM Token-Ring Network and that provides both subarea and peripheral node data link control (DLC) services in the SNA network.

NDF. NCP/EP definition facility.

negative response (NR). In SNA, a response indicating that a request did not arrive successfully or was not processed successfully by the receiver. Contrast with *positive response.*

NETID. Network identifier.

NetView-NetView task (NNT). The task under which a cross-domain NetView operator session runs. See *operator station task.*

NetView Performance Monitor (NPM). An IBM licensed program that collects, monitors, analyzes, and displays data relevant to the performance of a VTAM telecommunication network. It runs as an online VTAM application program.

network accessible unit (NAU). A logical unit (LU), physical unit (PU), control point (CP), or system services control point (SSCP). It is the origin or the destination of information transmitted by the path control network. Synonymous with *network addressable unit.*

network address. (1) In a subarea network, an address, consisting of subarea and element fields, that identifies a link, link station, physical unit, logical unit, or system services control point. Subarea nodes use network addresses; peripheral nodes use local addresses or local-form session identifiers (LFSIDs). The boundary function in the subarea node to which a peripheral node is attached transforms local addresses or LFSIDs to network addresses and vice versa. Contrast with *network name.* (2) According to ISO 7498-3, a name, unambiguous within the OSI environment, that identifies a set of network service access points.

network addressable unit (NAU). Synonym for *network accessible unit.*

network control (NC). In SNA, a request/response unit (RU) category used for requests and responses exchanged between physical units (PUs) for such purposes as activating and deactivating explicit and virtual routes and sending load modules to adjust peripheral nodes. See also *data flow control, function management data,* and *session control.*

network control mode. The mode in which a network control program can direct a communication controller to perform such activities as polling, device addressing, dialing, and answering. See also *emulation mode.*

network control program. A program, generated by the user from a library of IBM-supplied modules, that controls the operation of a communication controller.

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

network directory database. Synonym for *distributed directory database.*

network management. The process of planning, organizing, and controlling a communication-oriented data processing or information system.

network management vector transport (NMVT). A management services request/response unit (RU) that flows over an active session between physical unit management services and control point management services (SSCP-PU session).

network name. (1) The symbolic identifier by which end users refer to a network accessible unit, a link, or a link station within a given subnetwork. In APPN networks, network names are also used for routing purposes. Contrast with *network address*. (2) In a multiple-domain network, the name of the APPL statement defining a VTAM application program. The network name must be unique across domains. Contrast with *ACB name*. See *uninterpreted name*.

network node (NN). Synonym for *Advanced Peer-to-Peer Networking (APPN) network node*.

network-node domain. An APPN network-node control point, its attached links, the network resources for which it answers directory search requests (namely, its local LUs and adjacent LEN end nodes), the adjacent APPN end nodes with which it exchanges directory search requests and replies, and other resources (such as a local storage device) associated with its own node or an adjacent end node for which it provides management services.

network operator. (1) A person who controls the operation of all or part of a network. (2) In a multiple-domain network, a person or program responsible for controlling all domains. Contrast with *domain operator*.

network performance analyzer (NPA). A function of NCP that collects performance data about devices. The data is recorded by NPM.

Network Routing Facility (NRF). An IBM licensed program that resides in NCP. NRF provides a path for routing messages between terminals and routes messages over this path without going through the host processor.

network services. (1) The services within network accessible units that control network operation through SSCP-SSCP, SSCP-PU, SSCP-LU, and CP-CP sessions. (2) The session services (directory and route-selection functions) and management services provided by an APPN network-node control point to its domain.

Network Terminal Option (NTO). An IBM licensed program, used in conjunction with NCP, that allows certain non-SNA devices to participate in sessions with SNA application programs in the host processor. When data is sent from a non-SNA device to the host processor, NTO converts non-SNA protocol to SNA protocol; and when data is sent from the host processor to the non-SNA device, NTO converts SNA protocol to non-SNA protocol.

NMVT. Network management vector transport.

NN. Network node.

no response. In SNA, a protocol requested in the form-of-response-requested field of the request header that directs the receiver of the request not to return any response, regardless of whether or not the request is received and processed successfully. Contrast with *definite response* and *exception response*.

node. (1) In a network, a point at which one or more functional units connect channels or data circuits. (I) (2) Any device, attached to a network, that transmits and receives data. (3) An endpoint of a link or a junction common to two or more links in a network. Nodes can be processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities. (4) In VTAM, a point in a network defined by a symbolic name. See *major node* and *minor node*. (5) In NETDA/2, a combination of hardware, software, and microcode that can generate message traffic, receive and process message traffic, or receive and relay message traffic.

nonswitched line. A telecommunication line on which connections do not have to be established by dialing. Contrast with *switched line*.

notification. (1) An unscheduled, spontaneously generated report of an event that has occurred. (2) In OSI management, information emitted by a managed object relating to an event that has occurred within the managed object.

NOTIFY. A network services request that is sent by a system services control point (SSCP) to a logical unit (LU) to inform the LU of the status of a procedure requested by the LU.

NPA. Network performance analyzer.

NPM. NetView Performance Monitor.

NPSI. X.25 NCP Packet Switching Interface.

NRF. Network Routing Facility.

NTO. Network Terminal Option.

NTRI. NCP/Token-Ring interconnection.

O

OAF. Origin address field.

OAF'. Origin address field prime.

operand. (1) An entity on which an operation is performed. (I) (2) That which is operated upon. An operand is usually identified by an address part of an instruction. (A) (3) Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor. (4) An expression to whose value an operator is applied. See also *definition statement*, *keyword*, *keyword parameter*, and *parameter*.

operator. (1) In a language statement, the lexical entity that indicates the action to be performed on operands. See also *definition statement*. (2) A person or program responsible for managing activities controlled by a given piece of software such as MVS, the NetView program, or IMS. See *logged-on operator* and *network operator*. See also *autotask* and *operator station task*. (3) A person who operates a device. (4) A person who keeps a system running.

operator station task (OST). The NetView task that establishes and maintains the online session with the network operator. There is one operator station task for each network operator who logs on to the NetView program. See *NetView-NetView task*.

origin address field (OAF). In SNA, a field in a FID0 or FID1 transmission header that contains the address of the originating network accessible unit (NAU). Contrast with *destination address field*. See also *format identification (FID) field* and *local session identification (LSID)*.

P

pacing. A technique by which a receiving component controls the rate of transmission of a sending component to prevent overrun or congestion. See *session-level pacing*, *send pacing*, and *virtual route (VR) pacing*. See also *flow control*.

pacing response. In SNA, an indicator that signifies the readiness of a receiving component to accept another pacing group. The indicator is carried in a response header (RH) for session-level pacing and in a transmission header (TH) for virtual route pacing.

packet level. (1) The packet format and control procedures for exchange of packets containing control information and user data between data terminal equipment (DTE) and data circuit-terminating equipment (DCE). See also *data link level*, *higher level*, and *physical level*. (2) A part of Recommendation X.25 that defines the protocol for establishing logical connections between two DTEs and for transferring data on these connections.

packet mode operation. Synonym for *packet switching*.

packet switching. (1) The process of routing and transferring data by means of addressed packets so that a channel is occupied only during transmission of a packet. On completion of the transmission, the channel is made available for transfer of other packets. (I) (2) Synonymous with *packet mode operation*. See also *circuit switching*.

parallel transmission groups. Multiple transmission groups between adjacent nodes, with each group having a distinct transmission group number.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (I) (A) (2) In Basic CUA architecture, a variable used in conjunction with a command to affect its result. (3) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (4) Data passed to a program or procedure by a user or another program, namely as an operand in a language statement, as an item in a menu, or as a shared data structure. See also *keyword*, *keyword parameter*, and *operand*.

parameters. In NETDA/2, the set of restrictions that affect only the output of a network design. A change in a parameter value does not change the input to the network design. Contrast with *constraints*.

partitioned emulation programming (PEP) extension. A function of a network control program that enables a communication controller to operate some telecommunication lines in network control mode while simultaneously operating others in emulation mode.

path. (1) In a network, any route between any two nodes. A path may include more than one branch. (T) (2) The series of transport network components (path control and data link control) that are traversed by the information exchanged between two network accessible units. See also *explicit route (ER)*, *route extension*, and *virtual route (VR)*. (3) In VTAM when defining a switched major node, a potential dial-out port that can be used to reach that node. (4) In the NetView/PC program, a complete line in a configuration that contains all of the resources in the service point command service (SPCS) query link configuration request list.

path control (PC). The function that routes message units between network accessible units in the network and provides the paths between them. It converts the basic information units (BIUs) from transmission control (possibly segmenting them) into path information units (PIUs) and exchanges basic transmission units containing one or more PIUs with data link control. Path control differs by node type: some nodes (APPN nodes, for example) use locally generated session identifiers for routing, and others (subarea nodes) use network addresses for routing.

path information unit (PIU). A message unit consisting of a transmission header (TH) alone, or a TH followed by a basic information unit (BIU) or a BIU segment. See also *transmission header.*

PCF. Primary control field.

PCID. Procedure-correlation identifier.

pending active session. In VTAM, the state of an LU-LU session recorded by the system services control point (SSCP) when it finds both logical units (LUs) available and has sent a CINIT request to the primary logical unit (PLU) of the requested session.

PEP. Partitioned emulation programming.

performance error. Synonym for *temporary error.*

peripheral link. In SNA, a link between a subarea and a peripheral node. See also *route extension (REX).*

peripheral logical unit (LU). In SNA, a logical unit in a peripheral node.

peripheral node. A node that uses local addresses for routing and therefore is not affected by changes in network addresses. A peripheral node requires boundary-function assistance from an adjacent subarea node. A peripheral node can be a type 1, 2.0, or 2.1 node connected to a subarea boundary node.

peripheral path control. The function in a peripheral node that routes message units between units with local addresses and provides the paths between them. See *path control* and *subarea path control.* See also *boundary function, peripheral node,* and *subarea node.*

peripheral PU. In SNA, a physical unit (PU) in a peripheral node.

permanent error. An error that cannot be resolved by error recovery programs. Contrast with *temporary error.*

physical circuit. A circuit established without multiplexing. See also *data circuit.* Contrast with *virtual circuit.*

physical level. In X.25, the mechanical, electrical, functional, and procedural media used to activate, maintain, and deactivate the physical link between the data terminal equipment (DTE) and the data circuit-terminating equipment (DCE). See *data link level* and *packet level.*

physical unit (PU). The component that manages and monitors the resources (such as attached links and adjacent link stations) associated with a node, as requested by an SSCP via an SSCP-PU session. An SSCP activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links. This term applies to type 2.0, type 4, and type 5 nodes only. See also *peripheral PU* and *subarea PU.*

physical unit (PU) services. In SNA, the components within a physical unit (PU) that provide configuration services and maintenance services for SSCP-PU sessions. See also *logical unit (LU) services.*

PIU. Path information unit.

PLU. Primary logical unit.

pointer. (1) A data element that indicates the location of another data element. (T) (2) An identifier that indicates the location of an item of data. (A)

polling. (1) On a multipoint connection or a point-to-point connection, the process whereby data stations are invited, one at a time, to transmit. (I) (2) Interrogation of devices for such purposes as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (A)

port. (1) An access point for data entry or exit. (2) A connector on a device to which cables for other devices such as display stations and printers are attached. Synonymous with *socket.* (3) The representation of a physical connection to the link hardware. A port is sometimes referred to as an adapter; however, there can be more than one port on an adapter. There may be one or more ports controlled by a single DLC process. (4) In the Internet suite of protocols, a 16-bit number used to communicate between TCP or the User Datagram Protocol (UDP) and a higher-level protocol or application. Some protocols, such as File Transfer Protocol (FTP) and Simple Mail Transfer Protocol (SMTP), use the same well-known port number in all TCP/IP implementations. (5) An abstraction used by transport protocols to distinguish among multiple destinations within a host machine.

positional operand. An operand in a language statement that has a fixed position. See also *definition statement.* Contrast with *keyword operand.*

positive response. In SNA, a response indicating that a request was received and processed. Contrast with *negative response*.

POST. Power-on self test.

primary logical unit (PLU). In SNA, the logical unit (LU) that sends the BIND to activate a session with its partner LU. Contrast with *secondary logical unit*.

procedure-correlation identifier (PCID). In SNA, a value used to correlate all requests and replies associated with a given procedure.

protection key. An indicator that appears in the current program status word whenever an associated task has control of the system. This indicator must match the storage keys of all main storage blocks that the task is to use.

protocol. (1) A set of semantic and syntactic rules that determine the behavior of functional units in achieving communication. (I) (2) In Open Systems Interconnection architecture, a set of semantic and syntactic rules that determine the behavior of entities in the same layer in performing communication functions. (T) (3) In SNA, the meanings of, and the sequencing rules for, requests and responses used for managing the network, transferring data, and synchronizing the states of network components. Synonymous with *line control discipline* and *line discipline*. See *bracket protocol* and *link protocol*.

PU. Physical unit.

Q

quiesce. (1) To end a process by allowing operations to complete normally. (2) In a VTAM application program, for one node to stop another node from sending synchronous-flow messages.

R

RAS. Reliability, availability, and serviceability.

real address. The address by which a logical unit (LU) is known within the SNA network in which it resides.

receive not ready (RNR). In communications, a data link command or response that indicates a temporary condition of being unable to accept incoming frames.

receive not ready (RNR) packet. See *RNR packet*.

receive pacing. In SNA, the pacing of message units being received by a component. See also *send pacing*.

RECMS. Record maintenance statistics.

record maintenance statistics (RECMS). An SNA error event record built from an NCP or line error and sent unsolicited to the host.

reentrant. The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks.

REGS. Registers.

request header (RH). The control information that precedes a request unit (RU). See also *request/response header (RH)*.

request unit (RU). A message unit that contains control information, end-user data, or both.

request/response header (RH). Control information associated with a particular RU. The RH precedes the request/response unit (RU) and specifies the type of RU (request unit or response unit).

request/response unit (RU). A generic term for a request unit or a response unit. See *request unit (RU)* and *response unit (RU)*.

reset. On a virtual circuit, reinitialization of data flow control. At reset, all data in transit are eliminated.

resource. (1) Any facility of a computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In the NetView program, any hardware or software that provides function to the network.

resource resolution table (RRT). In NPM, this table contains the names of network resources for which data is to be collected. The NPM RRT corresponds with an NCP and is built by NPMGEN from an NCP Stage I and an NCP RRT.

response. In data communication, a reply represented in the control field of a response frame. It advises the primary or combined station of the action taken by the secondary or other combined station to one or more commands. See also *command*.

response header (RH). A header, optionally followed by a response unit (RU), that indicates whether the response is positive or negative and that may contain a pacing response. See also *negative response*, *pacing response*, and *positive response*.

response unit (RU). A message unit that acknowledges a request unit. It may contain prefix information received in a request unit. If positive, the response unit may contain additional information (such as session

parameters in response to BIND SESSION). If negative, the response unit contains sense data defining the exception condition.

REX. Route extension.

RH. Request/response header.

RNAA. Request network address assignment.

RNR. Receive not ready.

RNR packet. A packet used by a data terminal equipment (DTE) or by a data circuit-terminating equipment (DCE) to indicate a temporary inability to accept additional packets for a virtual call or permanent virtual circuit.

route extension (REX). In SNA, the path control network components, including a peripheral link, that make up the portion of a path between a subarea node and a network addressable unit (NAU) in an adjacent peripheral node. See also *explicit route (ER)*, *path*, and *virtual route (VR)*.

router. (1) A computer that determines the path of network traffic flow. The path selection is made from several paths based on information obtained from specific protocols, algorithms that attempt to identify the shortest or best path, and other criteria such as metrics or protocol-specific destination addresses. (2) An attaching device that connects two LAN segments, which use similar or different architectures, at the reference model network layer. Contrast with *bridge* and *gateway*. (3) In OSI terminology, a function that determines a path by which an entity can be reached.

routing. (1) The process of determining the path to be used for transmission of a message over a network. (T) (2) The assignment of the path by which a message is to reach its destination. (3) In SNA, the forwarding of a message unit along a particular path through a network, as determined by parameters carried in the message unit, such as the destination network address in a transmission header.

RR. Receive ready.

RU. Request/response unit.

RU chain. In SNA, a set of related request/response units (RUs) that are consecutively transmitted on a particular normal or expedited data flow. The request RU chain is the unit of recovery: if one of the RUs in the chain cannot be processed, the entire chain is discarded. Each RU belongs to only one chain, which has a beginning and an end indicated by means of control bits in request/response headers within the RU chain. Each RU can be designated as first-in-chain (FIC), last-in-chain (LIC), middle-in-chain (MIC), or only-in-chain (OIC). Response units and expedited-flow request units are always sent as only-in-chain.

S

SAF. Source address field.

scanner. (1) A device that examines a spatial pattern one part after another, and generates analog or digital signals corresponding to the pattern. Scanners are often used in mark sensing, pattern recognition, or character recognition. (I) (A) (2) For the 3725 communication controller, a processor dedicated to controlling a small number of telecommunication lines. It provides the connection between the line interface coupler hardware and the central control unit.

scanner interface trace (SIT). A record of the activity within the communication scanner processor (CSP) for a specified data link between an IBM 3725 Communication Controller and a resource.

SCB. (1) Session control block. (2) String control byte.

SDB. Storage descriptor block.

SDLC. Synchronous Data Link Control.

SDT. Start data traffic.

secondary logical unit (SLU). In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. An LU may contain secondary and primary half-sessions for different active LU-LU sessions. Contrast with *primary logical unit (PLU)*.

secondary logical unit (SLU) key. A key-encrypting key used to protect a session cryptography key during its transmission to the secondary half-session.

segment. (1) In the IBM Token-Ring Network, a section of cable between components or devices. A segment may consist of a single patch cable, several patch cables that are connected, or a combination of building cable and patch cables that are connected. (2) In Internet communications, the unit of transfer between TCP functions in different machines. Each segment contains control and data fields; the current byte stream position and actual data bytes are identified along with a checksum to validate received data. (3) Synonym for *BIU segment*. (4) See *link connection segment*.

segmentation. A process by which path control (PC) divides basic information units (BIUs) into smaller units, called BIU segments, to accommodate smaller buffer sizes in adjacent nodes. Both segmentation and segment assembly are optional PC features. The

support for either or both is indicated in the BIND request and response.

send pacing. In SNA, pacing of message units that a component is sending. See also *receive pacing*.

sequence number. (1) In communications, a number assigned to a particular frame or packet to control the transmission flow and receipt of data. (2) A numerical value assigned by VTAM to each message exchanged between two nodes. The value (one for messages sent from the application program to the logical unit and another for messages sent from the logical unit to the application program) increases by one for each successive message transmitted unless it is reset by the application program with a set and test sequence numbers (STSN) indicator.

service point (SP). An entry point that supports applications that provide network management for resources not under the direct control of itself as an entry point. Each resource is either under the direct control of another entry point or not under the direct control of any entry point. A service point accessing these resources is not required to use SNA sessions (unlike a focal point). A service point is needed when entry point support is not yet available for some network management function.

session activation request. In SNA, a request that activates a session between two network accessible units (NAUs) and specifies session parameters that control various protocols during session activity; for example, BIND and ACTPU. Contrast with *session deactivation request*.

session control (SC). In SNA, either of the following:

- One of the components of transmission control. Session control is used to purge data flowing in a session after an unrecoverable error occurs, to resynchronize the data flow after such an error, and to perform cryptographic verification.
- A request unit (RU) category used for requests and responses exchanged between the session control components of a session and for session activation and deactivation requests and responses.

session control block (SCB). In NPM, control blocks in common storage area for session collection.

session data. Data about a session, collected by the NetView program, that consists of session awareness data, session trace data, and session response time data.

session deactivation request. In SNA, a request that deactivates a session between two network accessible units (NAUs); for example, UNBIND and DACTPU. Synonymous with *generic unbind*. Contrast with *session activation request*.

session-level pacing. A flow control technique that permits a receiving half-session or session connector to control the data transfer rate (the rate at which it receives request units) on the normal flow. It is used to prevent overloading a receiver with unprocessed requests when the sender can generate requests faster than the receiver can process them. See *pacing* and *virtual route pacing*.

session partner. In SNA, one of the two network accessible units (NAUs) having an active session.

SIT. Scanner interface trace.

SLU. Secondary logical unit.

SNA. Systems Network Architecture.

SNA network. The part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network accessible units (NAUs), boundary function, gateway function, and intermediate session routing function components; and the transport network.

SNA network interconnection (SNI). The connection, by gateways, of two or more independent SNA networks to allow communication between logical units in those networks. The individual SNA networks retain their independence.

SNI. SNA network interconnection.

socket. The abstraction provided by the University of California's Berkeley Software Distribution (commonly called Berkeley UNIX or BSD UNIX) that serves as an endpoint for communication between processes or applications.

SP. Service point.

span. In the NetView program, a user-defined group of network resources within a single domain. Each major or minor node is defined as belonging to one or more spans. See also *span of control*.

span of control. The total network resources over which a particular network operator has control. All the network resources listed in spans associated through profile definition with a particular network operator are within that operator's span of control.

SS. (1) Start-stop. (2) Session services.

SSCP. System services control point.

SSCP-dependent LU. An LU that requires assistance from a system services control point (SSCP) in order to initiate an LU-LU session. It requires an SSCP-LU session.

SSCP ID. In SNA, a number that uniquely identifies a system services control point (SSCP). The SSCP ID is used in session activation requests sent to physical units (PUs) and other SSCPs.

SSCP-LU session. In SNA, a session between a system services control point (SSCP) and a logical unit (LU). The session enables the LU to request the SSCP to help initiate LU-LU sessions.

SSCP-PU session. In SNA, a session between a system services control point (SSCP) and a physical unit (PU); SSCP-PU sessions allow SSCPs to send requests to and receive status information from individual nodes in order to control the network configuration.

SSP. System Support Programs.

ST. Session configuration screen abbreviation.

stage. A program that processes messages in a NetView pipeline. Stages send messages to each other serially.

start-stop (SS) transmission. (1) Asynchronous transmission such that each group of signals representing a character is preceded by a start signal and is followed by a stop signal. (T) (A) (2) Asynchronous transmission in which a group of bits is (a) preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character, and (b) followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending reception of the next character. See also *binary synchronous transmission* and *synchronous data link control*.

stream. (1) To send data from one device to another. (2) See *data stream*.

subarea. A portion of the SNA network consisting of a subarea node, attached peripheral nodes, and associated resources. Within a subarea node, all network accessible units (NAUs), links, and adjacent link stations (in attached peripheral or subarea nodes) that are addressable within the subarea share a common subarea address and have distinct element addresses.

subarea address. A value in the subarea field of the network address that identifies a particular subarea. See also *element address*.

subarea node (SN). A node that uses network addresses for routing and maintains routing tables that reflect the configuration of the network. Subarea nodes can provide gateway function to connect multiple subarea networks, intermediate routing function, and boundary function support for peripheral nodes. Type 4 and type 5 nodes can be subarea nodes.

subarea path control. The function in a subarea node that routes message units between network accessible units (NAUs) and provides the paths between them. See *path control* and *peripheral path control*. See also *boundary function*, *peripheral node*, and *subarea node*.

subarea PU. In SNA, a physical unit (PU) in a subarea node.

subvector. A subcomponent of the NMVT major vector.

supervisor. The part of a control program that coordinates the use of resources and maintains the flow of processing unit operations.

supervisor call (SVC). A request that serves as the interface into operating system functions, such as allocating storage. The SVC protects the operating system from inappropriate user entry. All operating system requests must be handled by SVCs.

SVC. (1) Supervisor call. (2) Switched virtual circuit.

switchback. In an IBM 3745 Communication Controller with twin central control units (CCUs) in backup mode, the process by which, after fallback, buses are moved back to the central control unit (CCU) that originally operated them.

switched virtual circuit (SVC). An X.25 circuit that is dynamically established when needed. The X.25 equivalent of a switched line.

Synchronous Data Link Control (SDLC). A discipline conforming to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute (ANSI) and High-level Data Link Control (HDLC) of the International Organization for Standardization, for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half-duplex over switched or nonswitched links. The configuration of the link connection may be point-to-point, multipoint, or loop. (I) Contrast with *binary synchronous communication (BSC)*.

system restart. Synonym for *initial program load (IPL)*.

system services control point (SSCP). A component within a subarea network for managing the configura-

tion, coordinating network operator and problem determination requests, and providing directory services and other session services for end users of the network. Multiple SSCPs, cooperating as peers with one another, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its own domain.

system services control point (SSCP) domain. The system services control point, the physical units (PUs), the logical units (LUs), the links, the link stations, and all the resources that the SSCP has the ability to control by means of activation and deactivation requests.

system slowdown. A network control program mode of reduced operation invoked when buffer availability drops below a threshold level. The network control program limits the amount of new data that the system accepts while continuing normal output activity.

system startup. Synonym for *initial program load (IPL).*

System Support Programs (SSP). An IBM licensed program, made up of a collection of utilities and small programs, that supports the operation of the NCP.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. The layered structure of SNA allows the ultimate origins and destinations of information, that is, the end users, to be independent of and unaffected by the specific SNA network services and facilities used for information exchange.

T

TAB. Terminal anchor block.

takeover. The process by which the failing active subsystem is released from its extended recovery facility (XRF) sessions with terminal users and replaced by an alternate subsystem. See *resource takeover.*

TAP. Synonym for *ACF/TAP.*

task panel. Online display from which you communicate with the program in order to accomplish the program's function, either by selecting an option provided on the panel or by entering an explicit command. See *help panel.*

telecommunication line. (1) The portion of a data circuit external to a data circuit-terminating equipment (DCE) that connects the DCE to a data-switching exchange (DSE), that connects a DCE to one or more other DCEs, or that connects a DSE to another DSE. (T) (2) Any physical medium, such as a wire or microwave beam, that is used to transmit data. Synonymous with *transmission line.*

temporary error. A resource failure that can be resolved by error recovery programs. Synonymous with *performance error.* Contrast with *permanent error.*

TERMINATE. In SNA, a request unit that is sent by a logical unit (LU) to its system services control point (SSCP) to cause the SSCP to start a procedure to end one or more designated LU-LU sessions.

TG. Transmission group.

TH. Transmission header.

threshold. (1) In the NetView program, a percentage value, set for a resource and compared to a calculated error-to-traffic ratio. (2) In NPM, high or low values supplied by the user to monitor data and statistics being collected. (3) In IBM bridge programs, a value set for the maximum number of frames that are not forwarded across a bridge due to errors, before a "threshold exceeded" occurrence is counted and indicated to network management programs. (4) An initial value from which a counter is decremented to 0, or a value to which a counter is incremented or decremented from an initial value.

TIC. Token-ring interface coupler.

token. (1) In a local area network, the symbol of authority passed successively from one data station to another to indicate the station temporarily in control of the transmission medium. Each data station has an opportunity to acquire and use the token to control the medium. A token is a particular message or bit pattern that signifies permission to transmit. (T) (2) In LANs, a sequence of bits passed from one device to another along the transmission medium. When the token has data appended to it, it becomes a frame.

token ring. (1) According to IEEE 802.5, network technology that controls media access by passing a token (special packet or frame) between media-attached stations. (2) A FDDI or IEEE 802.5 network with a ring topology that passes tokens from one attaching ring station (node) to another. (3) See also *local area network (LAN).*

token-ring interface coupler (TIC). An adapter that can connect a 3720, 3725, or 3745 Communication Controller to an IBM Token-Ring Network.

TR. Trace.

Trace Analysis Program (TAP). Synonym for *Advanced Communications Function for the Trace Analysis Program (ACF/TAP).*

transmission control unit (TCU). A communication control unit whose operations are controlled solely by programmed instructions from the computing system to which the unit is attached. No program is stored or executed in the unit. Examples are the IBM 2702 and 2703 Transmission Controls. Contrast with *communication controller.*

transmission group (TG). (1) A connection between adjacent nodes that is identified by a transmission group number. See also *parallel transmission groups.* (2) In a subarea network, a single link or a group of links between adjacent nodes. When a transmission group consists of a group of links, the links are viewed as a single logical link, and the transmission group is called a *multilink transmission group (MLTG).* A *mixed-media multilink transmission group (MMMLTG)* is one that contains links of different medium types (for example, token-ring, switched SDLC, nonswitched SDLC, and frame-relay links). (3) In an APPN network, a single link between adjacent nodes.

transmission group (TG) vector. A representation of an endpoint TG in a T2.1 network, consisting of two control vectors: the TG Descriptor (X'46') control vector and the TG Characteristics (X'47') control vector.

transmission header (TH). Control information, optionally followed by a basic information unit (BIU) or a BIU segment, that is created and used by path control to route message units and to control their flow within the network. See also *path information unit.*

transmission line. Synonym for *telecommunication line.*

transmission priority. A rank assigned to a message unit that determines its precedence for being selected by the path control component in each node along a route for forwarding to the next node in the route.

U

UNBIND. In SNA, a request to deactivate a session between two logical units (LUs). See also *session deactivation request.* Contrast with *BIND.*

uninterpreted name. In SNA, a character string that a system services control point (SSCP) can convert into the network name of a logical unit (LU). Typically, an uninterpreted name is used in a logon or Initiate request from a secondary logical unit (SLU) to identify the primary logical unit (PLU) with which the session is requested.

UP. Unnumbered poll.

user exit. (1) A point in an IBM-supplied program at which a user exit routine may be given control. (2) A programming service provided by an IBM software product that may be requested during the execution of an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

user exit routine. A user-written routine that receives control at predefined user exit points. User exit routines can be written in assembler or a high-level language.

user-written generation application. A user-written program that runs with the NCP/EP definition facility (NDF) during NCP generation. It processes definition statements and operands.

V

variable. (1) In the NetView command list language, a character string beginning with "&" that is coded in a command list and is assigned a value during execution of the command list. (2) In the Simple Network Management Protocol (SNMP), a match of an object instance name with an associated value.

vector. The MAC frame information field.

virtual circuit. (1) In packet switching, the facilities provided by a network that give the appearance to the user of an actual connection. (T) See also *data circuit.* Contrast with *physical circuit.* (2) A logical connection established between two DTEs.

virtual link. In Open Shortest Path First (OSPF), a point-to-point interface that connects border routers that are separated by a non-backbone transit area. Because area routers are part of the OSPF backbone, the virtual link connects the backbone. The virtual links ensure that the OSPF backbone does not become discontinuous.

virtual machine (VM). In VM, a functional equivalent of a computing system. On the 370 Feature of VM, a virtual machine operates in System/370 mode. On the ESA Feature of VM, a virtual machine operates in System/370, 370-XA, ESA/370, or ESA/390 mode. Each virtual machine is controlled by an operating system. VM controls the concurrent execution of multiple virtual machines on an actual processor complex.

Virtual Machine/Enterprise Systems Architecture (VM/ESA). An IBM licensed program that manages the resources of a single computer so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of a *real* machine.

Virtual Machine/Extended Architecture (VM/XA). An operating system that facilitates conversion to MVS/XA by allowing several operating systems (a production system and one or more test systems) to run simultaneously on a single 370-XA processor. The VM/XA Migration Aid has three components: the control program (CP), the conversational monitor system (CMS), and the dump viewing facility.

virtual route (VR). In SNA, either a) a logical connection between two subarea nodes that is physically realized as a particular explicit route or b) a logical connection that is contained wholly within a subarea node for intranode sessions. A virtual route between distinct subarea nodes imposes a transmission priority on the underlying explicit route, provides flow control through virtual route pacing, and provides data integrity through sequence numbering of path information units (PIUs). See also *explicit route (ER)*, *path*, and *route extension (REX)*.

virtual route (VR) pacing. In SNA, a flow control technique used by the virtual route control component of path control at each end of a virtual route to control the rate at which path information units (PIUs) flow over the virtual route. VR pacing can be adjusted according to traffic congestion in any of the nodes along the route. See also *pacing* and *session-level pacing*.

Virtual Storage Extended (VSE). An IBM licensed program whose full name is the Virtual Storage Extended/Advanced Function. It is a software operating system controlling the execution of programs.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VM. Virtual machine.

VM/ESA. Virtual Machine/Enterprise Systems Architecture.

VM/SP. Virtual Machine/System Product.

VM/XA. Virtual Machine/Extended Architecture.

VM/370 control program (CP). The component of VM/370 that manages the resources of a single computer with the result that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of an IBM System/370 computing system.

VR. Virtual route.

VSE. Virtual Storage Extended. Synonymous with *VSE/Advanced Functions*.

VSE/Advanced Functions. The basic operating system support needed for a VSE-controlled installation. Synonym for *VSE*.

VSE/ESA. Virtual Storage Extended/Enterprise Systems Architecture.

VSE/SP. Virtual Storage Extended/System Package.

VTAM. Virtual Telecommunications Access Method. Synonymous with *ACF/VTAM*.

W

WAN. Wide area network.

wide area network (WAN). (1) A network that provides communication services to a geographic area larger than that served by a local area network or a metropolitan area network, and that may use or provide public communication facilities. (T) (2) A data communications network designed to serve an area of hundreds or thousands of miles; for example, public and private packet-switching networks, and national telephone networks. Contrast with *local area network (LAN)* and *metropolitan area network (MAN)*.

workstation. (1) A functional unit at which a user works. A workstation often has some processing capability. (T) (2) One or more programmable or nonprogrammable devices that allow a user to do work. (3) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

wrap. In general, to go from the maximum to the minimum in computer storage. For example, the continuation of an operation from the maximum value in storage to the first minimal value.

X

X.25. An International Telegraph and Telephone Consultative Committee (CCITT) recommendation for the interface between data terminal equipment and packet-switched data networks. See also *packet switching*.

X.25 NCP Packet Switching Interface (NPSI). An IBM licensed program that allows SNA users to communicate over packet switching data networks that have interfaces complying with CCITT Recommendation X.25. It allows SNA programs to communicate with SNA or non-SNA equipment over such networks.

XA. Extended architecture.

XID. Exchange identification.

Bibliography

NCP, SSP, and EP Library

The following paragraphs briefly describe the library for NCP, SSP, and EP. The other publications dealing with the networking systems products—NTune, VTAM, NPSI, the NetView program, and NPM—are listed without the accompanying descriptions.

NCP V7R2, SSP V4R2, and EP R12 Library Directory (SC31-6259)

This book helps users locate information on a variety of NCP, SSP, and EP tasks. It also provides a high-level understanding of NCP, SSP, and EP and summarizes the changes to these products and to the library for NCP V7R2, SSP V4R2, and EP R12.

NCP V7R2 Migration Guide (SC31-6258)

This book helps users migrate an NCP generation definition from an earlier release to NCP V7R2. It also describes how to add new functions for NCP V7R2.

NCP, SSP, and EP Resource Definition Guide (SC31-6223)

This book helps users understand how to define NCP and EP (in the PEP environment) using SSP. It describes functions and resources and lists the definition statements and keywords that define those functions and resources.

NCP, SSP, and EP Resource Definition Reference (SC31-6224)

This book helps users code definition statements and keywords to define NCP and EP (in the PEP environment) using SSP. It also provides a quick reference of definition statement coding order and keyword syntax.

NCP, SSP, and EP Generation and Loading Guide (SC31-6221)

This book provides detailed explanations of how to generate and load NCP and EP (in the PEP environment) using SSP. It contains information for generating and loading under MVS, VM, and VSE.

NCP and SSP Customization Guide (LY43-0031)

This book helps users who are familiar with the internal logic of NCP and SSP to modify these products. It describes how to change NCP and SSP to support stations that IBM-supplied programs do not support.

NCP and SSP Customization Reference (LY43-0032)

This book supplements the *NCP and SSP Customization Guide*. It describes the resources and macroinstructions provided by IBM for customizing NCP and SSP.

NCP, SSP, and EP Messages and Codes (SC31-6222)

This book is a reference book of abend codes issued by NCP and EP in the PEP environment, and messages issued by the System Support Programs associated with NCP. This information is also available through the online message facility, an IBM OS/2 application available on diskette.

NCP, SSP, and EP Trace Analysis Handbook (LY43-0037)

This book describes how to use the trace analysis program and how to read trace analysis program output.

NCP, SSP, and EP Diagnosis Guide (LY43-0033)

This book helps users isolate and define problems in NCP and EP (in the PEP environment) using SSP. The primary purpose of the book is to help the user interact with the IBM Support Center to resolve a problem. In addition, it explains some of the diagnostic aids and service aids available with SSP.

NCP, SSP, and EP Diagnosis Aid (LK2T-1999, diskettes)

The Diagnosis Aid is an IBM OS/2 application used to diagnose NCP, SSP, and EP problems. This tool helps programmers and program support personnel who are responsible for isolating, diagnosing, and debugging problems in NCP and EP (in the PEP environment) using SSP. The Diagnosis Aid, available on diskette, provides online access to all the information contained in the *NCP, SSP, and EP Diagnosis Guide*, the *NCP and EP Reference Summary and Data Areas*, the *NCP, SSP, and EP Messages and Codes*, and the *NCP, SSP, and EP Trace Analysis Handbook*.

NCP and EP Reference (LY43-0029)

This book describes various aspects of the internal processing of NCP and EP in the PEP environment. It provides information for customization and diagnosis.

NCP and EP Reference Summary and Data Areas (LY43-0030)

This two-volume book provides quick access to often-used diagnostic and debugging information about NCP and EP in the PEP environment.

Other Networking Systems Products Libraries

The following publications provide cross-product information for NTune, VTAM, NPSI, NetView, and NPM. For detailed information about these products, refer to the library for each.

Networking Systems Library

The following list shows the publications in the Networking Systems library (this library currently contains information about NCP at the V7R1 level).

Planning for NetView, NCP, and VTAM (SC31-7122)

Planning for Integrated Networks (SC31-7123)

Planning Aids: Pre-Installation Planning Checklist for NetView, NCP, and VTAM (SX75-0092)

IBM Networking Systems Softcopy Collection Kit (CD-ROM, SK2T-6012)

IBM Online Libraries: Softcopy Collection Kit User's Guide (GC28-1700)

NTune Library

The following list shows the publications in the NTune library.

NTune User's Guide (SC31-6247)

NTuneNCP Reference (LY43-0035)

VTAM Library

The following list shows the publications in the VTAM V4R2 library.

VTAM Migration Guide (GC31-6491)

VTAM Release Guide (GC31-6492)

Estimating Storage for VTAM (SK2T-2007)

VTAM Network Implementation Guide (SC31-6494)

VTAM Resource Definition Reference (SC31-6498)

VTAM Resource Definition Samples (SC31-6499, book and diskettes)

VTAM Customization (LY43-0063)

VTAM Operation (SC31-6495)

VTAM Operation Quick Reference (SX75-0205)

Using IBM CommandTree/2 (SC31-7013)

VTAM Messages and Codes (SC31-6493)

VTAM Licensed Program Specifications (GC31-6490)

VTAM Programming (SC31-6496)

VTAM Programming Quick Reference (SX75-0206)

VTAM Programming for LU 6.2 (SC31-6497)

VTAM Diagnosis (LY43-0065)

VTAM Diagnosis Quick Reference (LX75-0204)

VTAM Data Areas for MVS (LY43-0064)

NPSI Library

The following list shows the publications in the NPSI Version 3 library.

X.25 NCP Packet Switching Interface General Information (GC30-3469)

X.25 NCP Packet Switching Interface Planning and Installation (SC30-3470)

X.25 NCP Packet Switching Interface Host Programming (SC30-3502)

X.25 NCP Packet Switching Interface Diagnosis, Customization, and Tuning (LY30-5610)

X.25 NCP Packet Switching Interface Data Areas (LY43-0034)

X.25 NCP Packet Switching Interface Master Index (GC31-6206)

NetView Library

The following list shows the publications in the NetView V2R4 library.

NetView General Information (GC31-7098)

Learning about NetView (SK2T-6017, diskettes)

Learning about NetView Graphic Monitor Facility (SK2T-6018, diskettes)

NetView Graphic Monitor Facility Reference Poster (SX75-0100)

NetView Automation Planning (SC31-7083)

NetView Storage Estimates (SK2T-6016, diskette for a PS/2 or a PS/55)

NetView Installation and Administration Guide (SC31-7084 for MVS)

NetView Installation and Administration Facility/2 Guide (or *NIAF/2 Guide*, SC31-7099)

NetView Administration Reference (SC31-7080)

NetView Bridge Implementation (SC31-6131)

NetView Tuning Guide (SC31-7079)

NetView Automation Implementation (LY43-0016)

NetView Customization Guide (SC31-7091)

NetView Customization: Writing Command Lists (SC31-7092)

NetView Customization: Using PL/I and C (SC31-7093)

NetView Customization: Using Assembler (SC31-7094)

NetView Operation (SC31-7086)

NetView Graphic Monitor Facility User's Guide (SC31-7089)

NetView Command Quick Reference (SX75-0090)

NetView Messages (SC31-7096)

NetView Resource Alerts Reference (SC31-7097)

NetView Application Programming Guide (SC31-7081)

NetView Resource Object Data Manager Programming Guide (SC31-7095)

NetView Problem Determination and Diagnosis (LY43-0101)

NPM Library

The following list shows the publications in the NPM V2 library.

NetView Performance Monitor at a Glance (GH19-6960)

NetView Performance Monitor Concepts and Planning (GH19-6961)

NetView Performance Monitor User's Guide (SH19-6962)

NetView Performance Monitor Messages and Codes (SH19-6966)

NetView Performance Monitor Graphic Subsystem (SH19-6967)

NetView Performance Monitor Installation and Customization (SH19-6964)

NetView Performance Monitor Reports and Record Formats (SH19-6965)

NetView Performance Monitor Diagnosis (LY19-6381)

NetView Performance Monitor Desk/2 User's Guide (SH19-6963)

Related Publications

The following publications, though not directly related to NCP, may be helpful in understanding your network.

Emulation Program Resource Definition and Diagnosis (SC31-6205)

Communication Controller Publications

Assembler Language Publications: The following publication describes the assembler language used with all IBM communication controllers.

3704/3705 Communications Controller Assembler Language (GC30-3003)

3745 Publications: The following list shows selected publications for the IBM 3745 Communication Controller.

IBM 3745 Communication Controller Introduction (GA33-0092)

IBM 3745 Communication Controller Configuration Program (GA33-0093)

IBM 3745 Communication Controller (All Models): Principles of Operation (SA33-0102)

SNA Publications

The following publications contain information on SNA.

Systems Network Architecture Technical Overview (GC30-3073)

Systems Network Architecture Format and Protocol Reference Manual: Management Services (SC30-3346)

Systems Network Architecture Formats (GA27-3136)

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Communicating Your Comments to IBM

Network Control Program
System Support Programs
Customization Guide

NCP Version 7 Release 2
SSP Version 4 Release 2

Publication No. LY43-0031-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:

  United States and Canada: **1-800-227-5088**
- If you prefer to send comments electronically, use this network ID:
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink*: **CIBMORCF at RALVM13**
  - Internet: **USIB2HPD@VNET.IBM.COM**

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Help us help you!

Network Control Program
System Support Programs
Customization Guide

NCP Version 7 Release 2
SSP Version 4 Release 2

Publication No. LY43-0031-01

We hope you find this publication useful, readable and technically accurate, but only you can tell us! Your comments and suggestions will help us improve our technical publications. Please take a few minutes to let us know what you think by completing this form.

Overall, how satisfied are you with the information in this book?	Satisfied	Dissatisfied
	☐	☐

How satisfied are you that the information in this book is:	Satisfied	Dissatisfied
Accurate	☐	☐
Complete	☐	☐
Easy to find	☐	☐
Easy to understand	☐	☐
Well organized	☐	☐
Applicable to your task	☐	☐

Specific Comments or Problems:

Please tell us how we can improve this book:

Thank you for your response. When you send information to IBM, you grant IBM the right to use or distribute the information without incurring any obligation to you. You of course retain the right to use the information in any way you choose.

Name

Address

Company or Organization

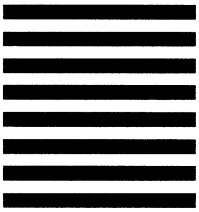
Phone No.

Help us help you!
LY43-0031-01

Cut
Alon

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Information Development
Department E15
International Business Machines Corporation
PO BOX 12195
RESEARCH TRIANGLE PARK NORTH CAROLINA 27709-9990

Fold and Tape | **Please do not staple** | Fold and Tape

Cut
Alon

Help us help you!

Network Control Program
System Support Programs
Customization Guide

NCP Version 7 Release 2
SSP Version 4 Release 2

Publication No. LY43-0031-01

We hope you find this publication useful, readable and technically accurate, but only you can tell us! Your comments and suggestions will help us improve our technical publications. Please take a few minutes to let us know what you think by completing this form.

Overall, how satisfied are you with the information in this book?	Satisfied	Dissatisfied
	☐	☐

How satisfied are you that the information in this book is:	Satisfied	Dissatisfied
Accurate	☐	☐
Complete	☐	☐
Easy to find	☐	☐
Easy to understand	☐	☐
Well organized	☐	☐
Applicable to your task	☐	☐

Specific Comments or Problems:

Please tell us how we can improve this book:

Thank you for your response. When you send information to IBM, you grant IBM the right to use or distribute the information without incurring any obligation to you. You of course retain the right to use the information in any way you choose.

Name

Address

Company or Organization

Phone No.

Help us help you!
LY43-0031-01

Cut
Alon

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

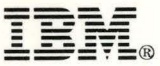
POSTAGE WILL BE PAID BY ADDRESSEE

Information Development
Department E15
International Business Machines Corporation
PO BOX 12195
RESEARCH TRIANGLE PARK NORTH CAROLINA 27709-9990

Fold and Tape **Please do not staple** Fold and Tape

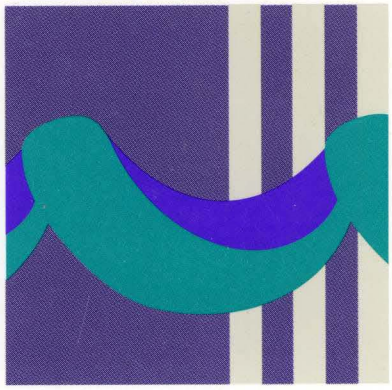
LY43-0031-01

Cut
Alor



File Number: S370/4300/30XX
Program Number: 5648-063
5665-041
5654-009
5686-064

Printed in U.S.A.



LY43-0031-01