

GC27-6998-0  
File No. S370-30

**Systems**

**VTAM Concepts and Planning**

**Virtual Telecommunications  
Access Method (VTAM)**

**DOS/VS  
OS/VS1  
OS/VS2**

**IBM**

GC27-6998-0  
File No. S370-30

**Systems**

## **VTAM Concepts and Planning**

**Virtual Telecommunications  
Access Method (VTAM)**

**DOS/VS  
OS/VS1  
OS/VS2**

**IBM**

**First Edition (May, 1974)**

This edition applies to the initial release of VTAM on DOS/VS and OS/VS, except where noted otherwise.

This edition is for planning purposes only. Specifications in this publication are subject to change, and such changes will be reported in subsequent revisions or technical newsletters.

Copies of this and other IBM publications can be obtained through your IBM representative or the IBM branch office serving your locality.

A form has been provided at the back of this publication for readers' comments. If this form has been removed, address comments to: IBM Corporation, Department 63T, Neighborhood Road, Kingston, New York 12401

## PREFACE

This publication provides a guide to planning for the installation and use of the Virtual Telecommunications Access Method (VTAM). The publication provides an introduction to VTAM, describes VTAM's major concepts and facilities, and provides detailed planning considerations. It is directed primarily to data processing managers and system programmers of installations that are considering installing a VTAM telecommunication system. This publication enables prospective users of VTAM to prepare for using VTAM. It is directed to both DOS/VS and OS/VS users.

This publication provides detailed information needed to plan a VTAM system and design VTAM application programs. (Refer to the publication *Introduction to VTAM*, GC27-6987, for a more general description of VTAM.)

This publication is organized as follows:

- Chapters 1 and 2 provide an introduction to VTAM and include descriptions of VTAM's major concepts and facilities. Complex concepts are introduced in these chapters and detailed in subsequent chapters.
- Chapters 3, 4, and 5 describe the primary interfaces to VTAM. Chapter 3 describes how VTAM can be used to define and tailor the telecommunication system. Chapter 4 describes how VTAM can be used to control the system, and Chapter 5 defines how VTAM is used by application programs for telecommunications.
- Chapter 6 describes VTAM's reliability, availability, and serviceability features.
- Chapter 7 provides hardware and software requirements for VTAM and discusses planning considerations for functions such as telecommunication security.

The reader should be familiar with basic teleprocessing concepts. These may be found in the publication *Introduction to Teleprocessing*, SC20-8095.

References are made in this publication to the *NCP Generation* publication. The most current edition of *NCP Generation* is titled:

*IBM 3704 and 3705 Communications Controllers Network Control Program/VS Generation and Utilities Guide and Reference Manual (for OS/VS TCAM Users)*, GC30-3007.

When used in reference to VTAM, this *NCP Generation* publication must be used only for planning purposes.

Other publications referred to in this publication are:

*DOS/VS Serviceability Aids and Debugging Procedures*, GC33-5380.

*Introduction to DOS/VS*, GC33-5370.

*Introduction to the IBM 3704 and 3705 Communications Controllers*, GA27-3051.

*OS/VS Dynamic Support System*, GC28-0640.

*OS/VS1 Planning and Use Guide*, GC24-5090.

*OS/VS1 Service Aids*, GC28-0665.

*OS/VS2 System Programming Library: Service Aids*, GC28-0633.

## Availability of VTAM Facilities

This publication applies to the initial release of VTAM on each of the three operating systems (DOS/VS, OS/VS1, and OS/VS2). In addition, the following VTAM facilities are described in this publication but are not planned to be distributed with the initial release of VTAM:

- Remote attachment of the IBM 3704 and 3705 Communications Controllers in a DOS/VS system.
- VTAM network-operator capability to change line-scheduling parameters in DOS/VS.

In addition the following teleprocessing subsystems will not be available with the initial release of VTAM: The IBM 3650 Retail Store System, the IBM 3660 Supermarket System, and the IBM 3790 Communication System.

# Contents

<b>Preface</b> . . . . .	iii
Availability of VTAM Facilities . . . . .	iv
<b>Chapter 1. Introduction to VTAM</b> . . . . .	1
What is VTAM? . . . . .	1
VTAM Provides System Coordination of Telecommunications Activity . . . . .	2
VTAM Provides a Telecommunication Base . . . . .	2
System Requirements for Using VTAM . . . . .	3
<b>Chapter 2. VTAM's Role in a Telecommunication System</b> . . . . .	5
Functions of VTAM . . . . .	5
Teleprocessing—An Overview . . . . .	5
A VTAM Telecommunication System . . . . .	6
Application Programs Using VTAM . . . . .	9
Communications Controllers in a VTAM Telecommunication Network . . . . .	9
Terminals in a VTAM Telecommunication Network . . . . .	10
Distributed Function . . . . .	12
Data Flow Through a VTAM Telecommunication System . . . . .	12
Sharing Resources—An Introduction . . . . .	13
Overview of VTAM . . . . .	13
Four Views of a VTAM System . . . . .	16
VTAM in Operation . . . . .	19
Installing a VTAM System . . . . .	19
Creating the System . . . . .	20
Controlling the Network . . . . .	20
Designing VTAM Application Programs . . . . .	20
Establishing Procedures for Using the System . . . . .	20
<b>Chapter 3. Creating a Telecommunication System with VTAM</b> . . . . .	23
VTAM Definition . . . . .	23
Generating VTAM . . . . .	25
Defining the Network . . . . .	25
Defining NCPs for Communications Controllers . . . . .	25
Defining Locally Attached 3270s . . . . .	26
Defining Application Programs . . . . .	29
Tailoring a VTAM System . . . . .	29
Defining VTAM Start Options . . . . .	30
Defining Logons . . . . .	31
The Network Solicitor . . . . .	36
Coding and Including Installation Exit-Routines . . . . .	39
VTAM Node Structure . . . . .	4
Node—A Definition . . . . .	42
Node Structure—Application Programs . . . . .	42
Node Structure—Local 3270s . . . . .	42
Node Structure—Local NCP . . . . .	43
Node Structure—Remote NCP . . . . .	44
Using the VTAM Node Structure . . . . .	44
Naming Nodes . . . . .	45
VTAM Buffering . . . . .	46
VTAM Buffers and Connected Terminals . . . . .	46
Specifying Buffer Sizes and Thresholds . . . . .	46
Controlling VTAM Buffering . . . . .	47
Characteristics of Logons . . . . .	47
Logon Types . . . . .	47
Automatic Logon . . . . .	48
Terminal-Initiated Logon . . . . .	48
Application-Program Logon . . . . .	49
Network-Operator Logon . . . . .	49
Comparing Types of Logons . . . . .	49
Logoffs . . . . .	50
<b>Chapter 4. Controlling a VTAM System</b> . . . . .	51
Levels of Control . . . . .	51
VTAM Commands . . . . .	51
Starting VTAM . . . . .	52
Starting VTAM in DOS/VS . . . . .	52
Starting VTAM in OS/VS . . . . .	52

Halting VTAM . . . . .	53
Orderly Closedown . . . . .	53
Quick Closedown . . . . .	53
Designing TPEND for the HALT Command . . . . .	53
Monitoring VTAM Status . . . . .	54
Activating and Deactivating Nodes . . . . .	55
Starting and Stopping an Application Program . . . . .	57
Activating and Deactivating Local 3270s . . . . .	57
Activating and Deactivating an NCP . . . . .	58
Activating and Deactivating Remote Attachments . . . . .	59
Activating and Deactivating Switched Networks . . . . .	60
Activating and Deactivating Teleprocessing Subsystems . . . . .	61
Special Considerations for Activation . . . . .	61
Initiating Requests for Connection . . . . .	62
Starting and Stopping VTAM Facilities . . . . .	62
Starting and Stopping the Network Solicitor . . . . .	63
Starting and Stopping Traces . . . . .	63
Starting the Trace-Print Utility Program . . . . .	63
Starting VTAM's Dump Utility Program . . . . .	64
Starting TOLTEP Testing . . . . .	64
Changing Line-Scheduling Specifications . . . . .	64
Considerations for Network-Operator Control . . . . .	65
<b>Chapter 5. VTAM Application Programs . . . . .</b>	<b>67</b>
A VTAM Application Program Overview . . . . .	67
The VTAM Application Program in Relation to the VTAM System . . . . .	67
The VTAM Application Program . . . . .	67
The Processing Part . . . . .	67
The Telecommunication Part . . . . .	68
VTAM . . . . .	69
The Network Control Program . . . . .	69
The Logical Unit . . . . .	69
The Terminal Operator . . . . .	69
A Summary of VTAM Macro Instructions . . . . .	69
Connection Macro Instructions . . . . .	70
Communication Macro Instructions . . . . .	70
Control-Block Macro Instructions . . . . .	70
Support Macro Instructions . . . . .	71
A Brief Comparison with TCAM and BTAM . . . . .	72
VTAM Compared with TCAM . . . . .	72
VTAM Compared with BTAM . . . . .	72
Application Program Concepts and Facilities . . . . .	72
Introduction to the Concepts and Facilities . . . . .	72
General Concepts and Facilities . . . . .	73
Overlapping VTAM Requests with Other Processing . . . . .	74
Application Program Exit-Routines . . . . .	77
Error Notification . . . . .	78
Opening the Application Program . . . . .	81
Connection . . . . .	81
Acceptance . . . . .	82
Acquisition . . . . .	83
Queuing Connection Requests . . . . .	84
Disconnection . . . . .	84
Communication . . . . .	86
Messages and Responses . . . . .	86
RRN and FME Responses . . . . .	89
Sequencing and Chaining . . . . .	92
Quiescing . . . . .	93
Facilities for Ensuring Orderly Communication . . . . .	94
Sequence Number Recovery . . . . .	96
Receiving Input . . . . .	100
Specific-Mode and Any-Mode . . . . .	101
Continue-Any and Continue-Specific Modes . . . . .	102
Identifying Logical Units . . . . .	102
Handling Overlength Input Data . . . . .	104
Communicating with the 3270 Information Display System . . . . .	105
Communicating with Start-Stop and BSC Terminals . . . . .	105

The VTAM Language . . . . .	109
Introduction to the VTAM Language . . . . .	110
The VTAM Macro Instructions . . . . .	110
The Connection Macro Instructions . . . . .	110
The Communication Macro Instructions . . . . .	111
The Control-Block Macro Instructions . . . . .	111
The Support Macro Instructions . . . . .	112
Relating VTAM Control Blocks to Executable Macro Instructions . . . . .	113
Opening The Application Program . . . . .	113
Connecting Logical Units . . . . .	114
Disconnecting Logical Units . . . . .	114
Communicating with Logical Units . . . . .	114
Handling Control Blocks, I/O Areas and Work Areas . . . . .	115
Coordinating I/O Activity . . . . .	115
Communicating with Start-Stop and BSC Terminals . . . . .	116
Distinguishing Between Logical Units and Start-Stop/BSC Terminals . . . . .	116
The LDO Control Block . . . . .	116
The CHANGE Macro Instruction . . . . .	116
The Basic-Mode Communication Macro Instructions . . . . .	117
Sample Programs . . . . .	117
Introduction to the Sample Programs . . . . .	117
Sample Program 1 . . . . .	118
The Logic of Sample Program 1 . . . . .	118
Sample Program 2 . . . . .	123
The Organization and Flow of Sample Program 2 . . . . .	126
The Logic of the 3600 I/O Routine . . . . .	131
The Logic of the 3600 Chaining-Output Routine . . . . .	133
The Logic of the 3270 I/O Routine . . . . .	136
The Logic of the RESP Exit-Routine . . . . .	137
The Logic of the DFASY Exit-Routine . . . . .	139
Choosing Programming Alternatives Discussed in the Sample Programs . . . . .	140
<b>Chapter 6. Reliability, Availability, Serviceability . . . . .</b>	<b>143</b>
VTAM's RAS Strategy . . . . .	143
VTAM's RAS Facilities . . . . .	144
Serviceability Aids . . . . .	144
Error Recording . . . . .	144
Traces . . . . .	145
Dumps . . . . .	146
Teleprocessing Online Test Executive Program (TOLTEP) . . . . .	147
Reliability and Availability Support . . . . .	148
Error Detection and Feedback . . . . .	149
NCP Initial Test . . . . .	149
Storage Management . . . . .	149
Error Recovery . . . . .	150
NCP Slowdown . . . . .	151
Configuration Restart . . . . .	151
<b>Chapter 7. VTAM Planning Considerations and Requirements . . . . .</b>	<b>153</b>
Machine Requirements . . . . .	153
CPU Support . . . . .	153
Locally Attached 3270s . . . . .	153
Requirements for Communications Controllers . . . . .	153
Remotely Attached Terminals . . . . .	154
Operating System Requirements . . . . .	155
Upward Compatibility . . . . .	155
Requirements for DOS/VS . . . . .	155
Requirements for OS/VS . . . . .	156
Network Control Program Requirements . . . . .	156
Introduction to the Network Control Program . . . . .	156
NCP Functions Required by VTAM . . . . .	156
Defining the NCP . . . . .	157
Support for Switched Networks . . . . .	159
Call-in Terminals . . . . .	160
Call-out Terminals . . . . .	161
Call-in/Call-out Terminals . . . . .	161
Partitioned Emulation Programming (PEP) Considerations . . . . .	161



VTAM Data Sets . . . . .	162
Data Sets for VTAM Under OS/VS . . . . .	162
Data Requirements for VTAM Under DOS/VS . . . . .	165
Sharing Telecommunication Resources . . . . .	166
Resources That Can Be Shared . . . . .	166
Sharing and Managing Resources . . . . .	166
Managing Resources Through VTAM Definition . . . . .	167
Managing Resources Through NCP Generation . . . . .	168
Managing Resources Through Application Programs . . . . .	168
Telecommunication Security Through VTAM . . . . .	169
Introducing Telecommunication Security . . . . .	169
Identification Verification . . . . .	170
Controlling Connections . . . . .	171
Authorization Exit-Routine . . . . .	171
Acquiring and Passing Connections . . . . .	172
Controlling Logon Requests . . . . .	172
Symbolic Names . . . . .	175
Controlling Access to VTAM . . . . .	175
Controlling the Use of VTAM Facilities . . . . .	175
Protecting Sensitive Data . . . . .	176
Other Telecommunication Access Methods . . . . .	176
TCAM Programs Under VTAM . . . . .	177
DOS/VS Coexistence . . . . .	178
OS/VS Coexistence . . . . .	178
<b>Appendix A. Local 3270 Support List . . . . .</b>	<b>181</b>
<b>Appendix B. Remotely Attached Terminals . . . . .</b>	<b>183</b>
Synchronous Data Link Control (SDLC) Devices . . . . .	183
Start-Stop Terminals . . . . .	184
Binary Synchronous Communications (BSC) Terminals . . . . .	186
<b>Appendix C. Remote Station Versus Remote Controller . . . . .</b>	<b>191</b>
<b>Appendix D. Summary of Application Program Indicators . . . . .</b>	<b>195</b>
<b>Glossary . . . . .</b>	<b>199</b>
<b>Index . . . . .</b>	<b>205</b>

## Figures

Figure 2-1. A VTAM Telecommunication System . . . . .	7
Figure 2-2. Nodes in a VTAM System . . . . .	8
Figure 2-3. Types of VTAM Terminals . . . . .	11
Figure 2-4. Correlation Between VTAM Logic Units and Teleprocessing Subsystems . . . . .	13
Figure 2-5. Data Flow Through a VTAM Telecommunication System . . . . .	14
Figure 2-6. Sharing Resources . . . . .	15
Figure 2-7. Four Views of a VTAM Telecommunication System . . . . .	17
Figure 3-1. Defining a VTAM System (Part 1 of 2) . . . . .	23
Figure 3-1. Defining a VTAM System (Part 2 of 2) . . . . .	24
Figure 3-2. Generating NCP Support in a VTAM Telecommunication System . . . . .	27
Figure 3-3. Grouping Locally Attached 3270s into Logical Sets . . . . .	28
Figure 3-4. Providing Control Information for Processing Logon Requests from Start-Stop and BSC Terminals . . . . .	33
Figure 3-5. Processing a Terminal-Initiated Logon with the Network Solicitor . . . . .	36
Figure 5-1. VTAM Application Programs in Relation to the Telecommunication System . . . . .	68
Figure 5-2. Relationship of VTAM's Control Blocks in an Application Program . . . . .	71
Figure 5-3. Relationship of TCAM to VTAM . . . . .	73
Figure 5-4. Major Similarities and Differences Between VTAM and BTAM Application Programs (Part 1 of 2) . . . . .	74
Figure 5-4. Major Similarities and Differences Between VTAM and BTAM Application Programs (Part 2 of 2) . . . . .	75
Figure 5-5. Processing Pattern for a Synchronous Request . . . . .	76
Figure 5-6. Processing Pattern When an ECB is Used with an Asynchronous Request . . . . .	77
Figure 5-7. Processing Pattern When an RPL Exit-Routine is Used with an Asynchronous Request . . . . .	78
Figure 5-8. A Possible Processing Pattern When Asynchronous Requests are Issued in RPL Exit-Routines . . . . .	79
Figure 5-9. Processing Pattern for Reporting Errors During an Asynchronous Operation . . . . .	80
Figure 5-10. Queued and Unqueued Connection Resuests . . . . .	85
Figure 5-11. Exchanging Messages and Responses . . . . .	87
Figure 5-12. Scheduled Output . . . . .	88
Figure 5-13. Responded Output . . . . .	89
Figure 5-14. Two VTAM Responses to Messages . . . . .	90
Figure 5-15. Two Possible Uses of the Response Types . . . . .	91
Figure 5-16. An Example of Message Chaining . . . . .	93
Figure 5-17. An Example of Quiesce Communication . . . . .	94
Figure 5-18. An Example of Change-Direction Communication . . . . .	96
Figure 5-19. An Example of Bracket Communication . . . . .	97
Figure 5-20. Indicators Used to Direct the Flow of Data . . . . .	98
Figure 5-21. An Example of Start-Data-Traffic and Clear Indicators . . . . .	99
Figure 5-22. Types of Information Exchanged Between an Application Program and Logical Unit . . . . .	101
Figure 5-23. Using a Combination of Any-Mode and Specific-Mode to Obtain Data . . . . .	103
Figure 5-24. Using the Continue-Any and Continue-Specific Modes to Handle Concurrent Inquiries . . . . .	104
Figure 5-25. Implicit and Explicit Solicitation in the Basic-Mode . . . . .	107
Figure 5-26. Basic-Mode Messages and Transmissions from an RJE Station Vary In Length . . . . .	109
Figure 5-27. The Logic of Sample Program 1 . . . . .	119
Figure 5-28. Hypothetical Network-Configuration for Sample Program 2 . . . . .	124
Figure 5-29. Organization and Flow of Sample Program 2 . . . . .	125
Figure 5-30. The Logic of the 3600 I/O Routine . . . . .	132
Figure 5-31. The Logic of the 3600 Chaining-Output Routine . . . . .	134
Figure 5-32. The Logic of the 3270 I/O Routine . . . . .	135
Figure 5-33. The Logic of the RESP Exit-Routine . . . . .	138
Figure 5-34. The Logic of the DFASY Exit-Routine . . . . .	139
Figure 7-1. Table of Data Sets Used by VTAM Under OS/VS (Part 1 of 2) . . . . .	163
Figure 7-1. Table of Data Sets Used by VTAM Under OS/VS (Part 2 of 2) . . . . .	164
Figure 7-2. Table of Files Used by VTAM Under DOS/VS . . . . .	165
Figure 7-3. VTAM's Use of Libraries Under DOS/VS . . . . .	166
Figure 7-4. Communications Controllers and Transmission Control Units in a Telecommunication Network . . . . .	177
Figure 7-5. DOS/VS Coexistence . . . . .	179
Figure 7-6. OS/VS Coexistence . . . . .	180
Figure B-1. Summary of Terminals That Can Operate in a VTAM Network . . . . .	190
Figure C-1. A Remotely Attached Communications Controllor . . . . .	193
Figure C-2. Communications Controllers Attached as Part of Remote Stations . . . . .	193



## CHAPTER 1. INTRODUCTION TO VTAM

This chapter summarizes the IBM Virtual Telecommunications Access Method (VTAM). It briefly tells what VTAM is and what it does.

### What is VTAM?

The IBM Virtual Telecommunications Access Method (VTAM) directs the transmission of data between application the connections between application programs and VTAM terminals. VTAM is a direct-control access method, and it enables application programs to communicate with VTAM terminals without concern for intermediate connections such as control units and telecommunication lines. *Note:* A VTAM terminal can be an application program in a teleprocessing subsystem (such as the IBM 3600 Finance Communication System), as well as terminals using start-stop or binary-synchronous-communications (BSC) line control.

VTAM performs the following services:

- Establishes, terminates, and controls access between application programs and terminals.
- Transfers data between application programs and terminals.
- Permits application programs to share communication lines, communications controllers, and terminals.
- Permits the operation of the telecommunication network to be monitored and altered.
- Permits the configuration of the telecommunication network to be changed while the network is being used.

Various users have access to these services:

- The application programmer can use the connection and data transfer services.
- The terminal operator and teleprocessing subsystems (such as the 3600 Finance Communication System) can use VTAM's services for logging onto application programs.
- The network operator can use VTAM's services for monitoring and controlling the telecommunication network.
- The system programmer can use VTAM's services for configuring the telecommunication network.

VTAM executes under the DOS/VS, OS/VS1, and OS/VS2 operating systems.

VTAM supports the local attachment of the IBM 3270 Information Display Station, and it uses the facilities of IBM 3704 or 3705 Communications Controllers with the network control program (NCP) in network control mode to control remotely attached devices. VTAM uses the NCP with or without the partitioned emulation programming (PEP) extension. It also supports the remotely attached communications controller.

VTAM supports the IBM teleprocessing subsystems that use synchronous data line control (SDLC). VTAM is designed to use the remote computing capability of these subsystems and the full-duplex capability of SDLC. VTAM also supports some of the terminals that use start-stop or BSC line disciplines.

VTAM allows application programs written for TCAM to be executed in a VTAM system, using VTAM facilities and resources; these application programs use VTAM through

TCAM. VTAM can coexist with BTAM; that is, application programs written for BTAM can be executed in a data processing system containing VTAM, but they must use a network of terminals separate from the VTAM network. VTAM and BTAM can share a 3704 or 3705 Communications Controller by using an NCP with the PEP extension.

## **VTAM Provides System Coordination of Telecommunication Activity**

VTAM assists in providing a system solution to telecommunication problems. Using the facilities of the operating system, of NCPs, and of teleprocessing subsystems, VTAM manages a telecommunication network and enables data to be transmitted between application programs in the central computer and remote locations.

VTAM supports the distribution of function outward from the central computer to devices such as the 3704 and 3705 Communications Controllers and SDLC cluster controllers (for example, the IBM 3601 Finance Controller). By distributing function, less of the central computer's capacity needs to be devoted to activities associated with controlling the telecommunication network and more of its capacity can be applied to the processing of data.

VTAM enables application programs in the central computer to communicate with application programs in SDLC cluster controllers. It provides a method of communication for the transmission of data independently of how devices are attached to the central computer and independently of the type of subsystem being used. This same method of communication can be used with IBM 3270 Information Display Systems; thus, 3270s can be treated, in many respects, like the application programs in SDLC cluster controllers.

In addition to enabling application programs in the central computer to communicate with applications in SDLC cluster controllers, VTAM enables these same application programs in the central computer to communicate with certain start-stop and BSC terminals. VTAM offers direct-control communication with the applications in the SDLC cluster controllers and with the start-stop and BSC terminals. If queued control is needed, applications in the central computer can use VTAM through TCAM. Communication with start-stop and BSC terminals is device-dependent.

In a VTAM system, the application program in the central computer:

- *Is independent* of line activities (such as line scheduling).
- *Is independent* of attachment concerns (such as whether a terminal is locally or remotely attached).
- *Is capable* of referencing terminals and application programs in SDLC cluster controllers symbolically.
- *Communicates* with terminals and application programs in SDLC cluster controllers on a real-time basis.

## **VTAM Provides a Telecommunication Base**

VTAM is designed to use advanced hardware and software including System/370 virtual storage, the IBM communications controllers, DOS/VS, OS/VS1, OS/VS2, and the teleprocessing subsystems that use the SDLC line discipline (such as the IBM 3600 Finance Communication System). With the IBM Virtual Storage Access Method (VSAM), VTAM can be used to provide a complementary data base/data communication facility. In addition to its primary role of data transmission, VTAM has features that establish it as a base for building telecommunication systems ranging in size from small to large. These features are:

- Sharing of network resources, which reduces line costs and makes more efficient use of the network.
- Concurrent execution of TCAM and VTAM application programs using the same telecommunication network.
- Services required for interactive applications (for example, online inquires and updates).
- Operation of the IBM 3704 and 3705 Communications Controllers to reduce the number of functions performed in the central computer for remote devices.
- Generation options for tailoring the telecommunication system to user's needs.
- Support for remotely attached central processing units (CPUs).
- Support for teleprocessing subsystems, such as the IBM 3600 Finance Communication System.
- Support for many different terminals which use different line disciplines (start/stop, binary synchronous control, and synchronous data link control).
- Reliability, availability, and serviceability aids to assist in reducing the incidence of errors in the telecommunication system, in reducing the impact of errors that do occur, and in maintaining the telecommunication system.

## **System Requirements for Using VTAM**

VTAM is a component of the DOS/VS, the OS/VS1, and the OS/VS2 operating systems. DOS/VS systems must have DOS/VS multiprogramming support.

The System/370 instruction set must include the Compare and Swap and the Compare Double and Swap instructions. These instructions are part of a hardware feature available on System/370 CPUs.

VTAM requires a 3704 or a 3705 in network control mode to support remotely attached devices. Appendixes A and B list devices that can be used by VTAM.



## CHAPTER 2. VTAM'S ROLE IN A TELECOMMUNICATION SYSTEM

This chapter discusses the role of VTAM in a telecommunication system and explains how VTAM relates to the other components in the system. Basic VTAM characteristics and concepts are also introduced.

### Functions of VTAM

VTAM manages the activities of a telecommunication system. It allocates resources and manages the flow of data between the central processor and the remote locations in the telecommunication system. To accomplish this, VTAM provides the following functions:

- *Initiation and termination:* VTAM enables an installation to define the telecommunication system and some of its characteristics. Once the system is defined, VTAM can be started and the system initialized. VTAM can also be used to shut down the telecommunication system in an orderly fashion.
- *Dynamic configuration:* VTAM enables the network operator to monitor the use of the resources within the telecommunication system and to alter the network as necessary.
- *Allocation:* VTAM controls the allocation of network resources. By owning and controlling all resources, VTAM provides a focal point within the system for controlling the network; yet, at the same time, it enables the installation to use its network efficiently.
- *I/O processing:* VTAM manages the transmission of data between application programs and terminals. It enables application programs and terminals to communicate with each other independently of how the terminal is connected to the central processing unit. VTAM also relies upon the distributed intelligence throughout the network (in communications controllers, cluster control units, etc.) to reduce the processing requirements in the central processing unit.
- *Reliability, availability, serviceability (RAS):* VTAM offers a design and facilities that reduce the incidence of problems in the telecommunication system, reduce the impact of errors that do occur, and assist in maintaining the telecommunication system.

These functions provide a flexible, dynamic telecommunication system—a system that is responsive to varying installation needs, that allows extensive sharing of network resources, and that provides central control for important activities such as telecommunication security.

Understanding how VTAM works requires familiarity with how the access method fits into a teleprocessing system and a telecommunication network. To provide that understanding is the purpose of the next two sections.

### Teleprocessing—An Overview

The purpose of a *teleprocessing system* is to make the power of a central computer available to a number of users working at remote locations. To achieve this aim, a teleprocessing system must do two main things:

- Transmit data between the central computer and the remote locations.
- Process data in the central computer.

VTAM is directly involved in the transmission of data. But the processing of data in the central computer is not a function of VTAM; instead, it is the responsibility of the application program using VTAM to transmit data.



That part of a teleprocessing system devoted to the transmission of data between the central computer and the remote locations is referred to as the *telecommunication system*. Figure 2-1 depicts the major elements in a VTAM telecommunication system.

The core of the telecommunication system is the central computer, which comprises a central processing unit (referred to as the *host CPU*), channels, and auxiliary storage. In a VTAM system, the CPU must be a System/370. The major components in the CPU are the operating system (DOS/VS, OS/VS1, or OS/VS2), VTAM, and the application programs that use VTAM.

Besides the central computer, the other major component of the system is the *telecommunication network*. The network is composed of communications controllers, telecommunication lines, control units, and terminals.

## A VTAM Telecommunication System

A telecommunication system can be viewed as a network of nodes coordinated by the telecommunication access method. (Nodes are addressable points in the telecommunication system, including application programs and terminals.) This network of nodes is shown in Figure 2-2.

An application program and a terminal can communicate with each other only through VTAM. To communicate, these two nodes must be connected. *Connection* is the allocation process by which VTAM establishes a path between an application program and a terminal. The *path* is composed of other nodes (including lines, communications controllers, and cluster control units) needed to transmit data between the application program and the terminal.

*Note:* For VTAM, a *cluster control unit* is one of the following: the IBM 2972 Station Control Unit, the IBM 3271 Control Unit, the IBM 3275 Display Station, and the control units for the teleprocessing subsystems (for example, the IBM 3601 Finance Communication Controller). The phrase *cluster control unit* is used in this publication when no distinction is needed between the BSC and the SDLC control units. When a reference is made to only the SDLC control units, the term *SDLC cluster controller* is used.

A terminal is connected to an application program for as long as explicitly requested by the application program, but intermediate nodes are allocated (but not connected) only for the duration of a single data-transfer operation (such as a single read or write operation). Consequently, intermediate nodes can be available simultaneously for paths to other terminals.

In providing the mechanism for transmitting data between application programs and terminals, VTAM removes much of the responsibility of network management from the application programs. The primary task involved in network management is resource allocation.

As network manager, VTAM handles the allocation of resources; that is, VTAM controls who uses what resources. The only aspects of resource allocation of direct concern to application programs are: (1) issuance of requests to have terminals connected to them and (2) acceptance of requests by terminals (logon) to be connected to an application program. Other aspects of allocation, including the control and use of intermediate nodes, are handled by VTAM. The number, type, and disposition of intermediate nodes are transparent to the terminals and to the application programs. (For more details on the concept of connection, see "Application Program Concepts and Facilities," in Chapter 5.)

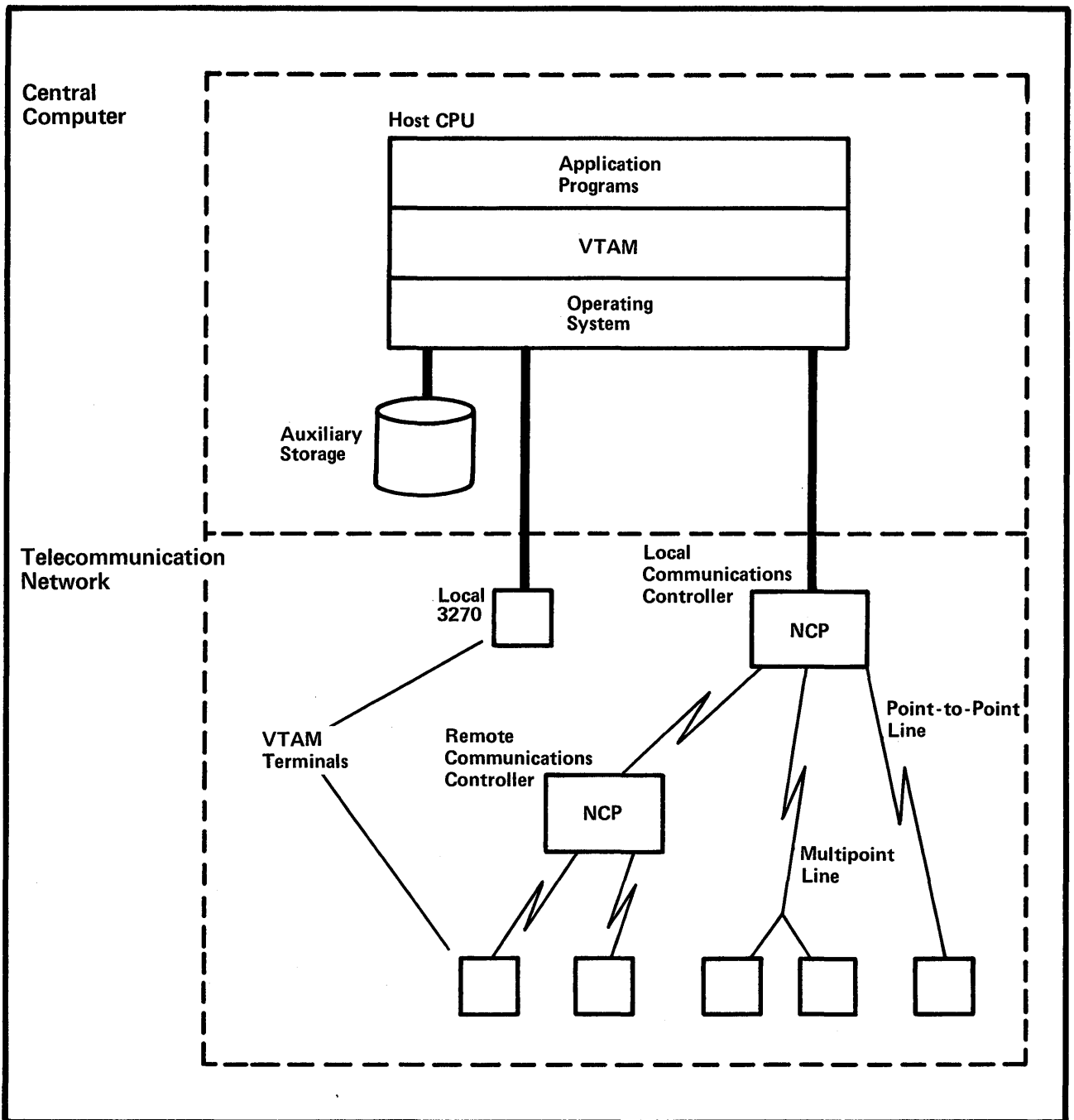


Figure 2-1. A VTAM Telecommunication System

The more significant nodes in a VTAM system include:

- Application programs.
- Network control programs (NCPs) for communications controllers.
- Terminals.

These nodes are discussed below. Additional details on VTAM nodes are provided under "VTAM Node Structure," in Chapter 3.

APPLICATION PROGRAMS IN HOST CPU

TELECOMMUNICATION NETWORK

Legend:

Nodes



VTAM Terminal



VTAM Application Program



Cluster Control Units and Communications Controllers

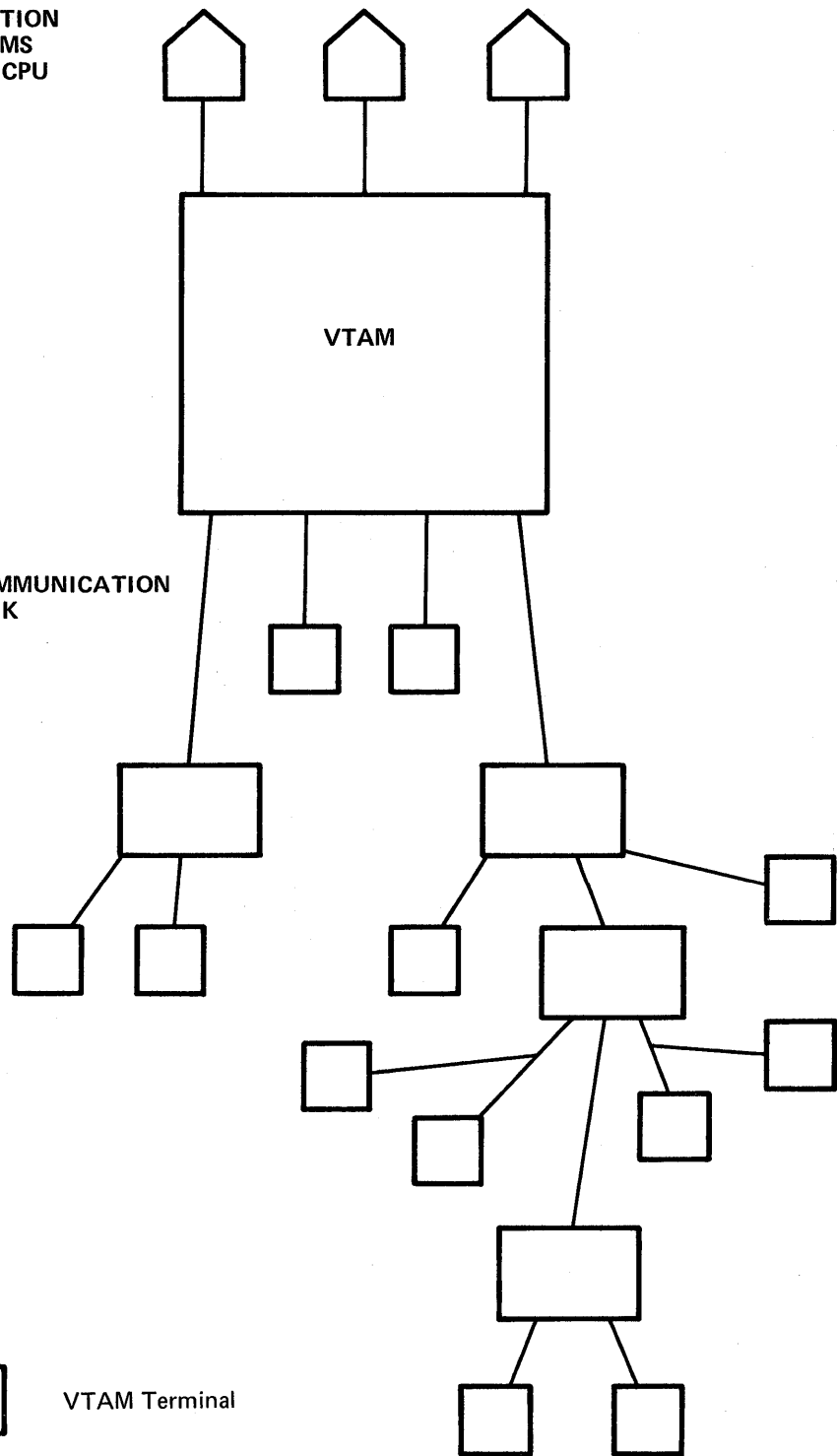


Figure 2-2. Nodes in a VTAM System

## ***Application Programs Using VTAM***

Each application program that uses VTAM establishes access to VTAM by a special control block. This control block, the access-method control block (ACB), is built by the application program.

The program must identify itself to VTAM prior to using any of VTAM's facilities; this identification is completed when the application program successfully opens its ACB (that is, initializes its ACB by issuing an OPEN macro instruction) and results in the application connecting to VTAM. As long as the ACB is open, the application program can request VTAM services. Such VTAM services can include:

- Allocating terminals to the application program.
- Transmitting data between a terminal and the application program.

A VTAM application program is any program that uses VTAM and its macro instructions to communicate with a terminal. The VTAM application program could be a program that both communicates with a terminal and processes its data. On the other hand, the VTAM application program could be an installation-written service program that handles the I/O requests of other application programs and does not process data. The use of the term *application program* throughout this book is not meant to imply one type of program over the other.

**Note:** Communication between application programs in the host CPU is not supported by VTAM; although such communication may be possible via other components of the operating system.

## ***Communications Controllers in a VTAM Telecommunication Network***

VTAM is designed to use the IBM 3704 and 3705 Communications Controllers. These controllers can be either *locally attached* (that is, connected to the host CPU via a data channel) or *remotely attached* (that is, connected to a locally attached communications controller via a telecommunication line).

Communications controllers link VTAM with the remote portions of the network and control the flow of information between terminals and VTAM.

The communications controllers are programmable devices; the program that controls these devices for VTAM is called the network control program/VS (referred to as *NCP* in this publication). The NCP has generation options that allows a communications controller to be operated in different modes. When operated in *network control mode*, the communication controller allows its network resources to be shared. Communications controllers can also be operated in *emulation mode*; when operated in this mode, they emulate the IBM 2701 Data Adapter Unit and the IBM 2702 and 2703 Transmission Control units. In addition, an NCP can be generated with the partitioned emulation programming (PEP) extension, which allows a communication controller to handle separate telecommunication lines in either network control or emulation mode at the same time. VTAM uses only the network control mode of the NCP, with or without PEP.

The NCP allows some functions previously performed entirely in the host CPU to be performed in the communications controller. Among the functions now provided by the communications controller are:

- Controlling lines.
- Controlling dynamic buffering.
- Deleting and inserting line control characters.
- Translating character codes.
- Detecting machine checks.

- Detecting permanent line errors.
- Gathering line statistics.
- Activating and deactivating lines.
- Closing down the network.
- Handling recoverable errors.
- Stamping dates and times.
- Providing error statistics to VTAM.

By performing these functions outside the host CPU and by allowing much of the network traffic to be controlled in the network, the NCP conserves central computing resources.

Although these activities are actually performed in the communications controllers, they are controlled by VTAM. For example, when the NCP is to deactivate a line, the command to deactivate comes from VTAM. VTAM, in turn, is controlled by the installation's definition of the system, by the network operator, and by the VTAM application programs.

### ***Terminals in a VTAM Telecommunication Network***

VTAM supports remotely attached terminals (that is, terminals connected to a communications controller via a telecommunication line) that use the following line disciplines:

- Start-stop.
- Binary synchronous communications (BSC).
- Synchronous data link control (SDLC).

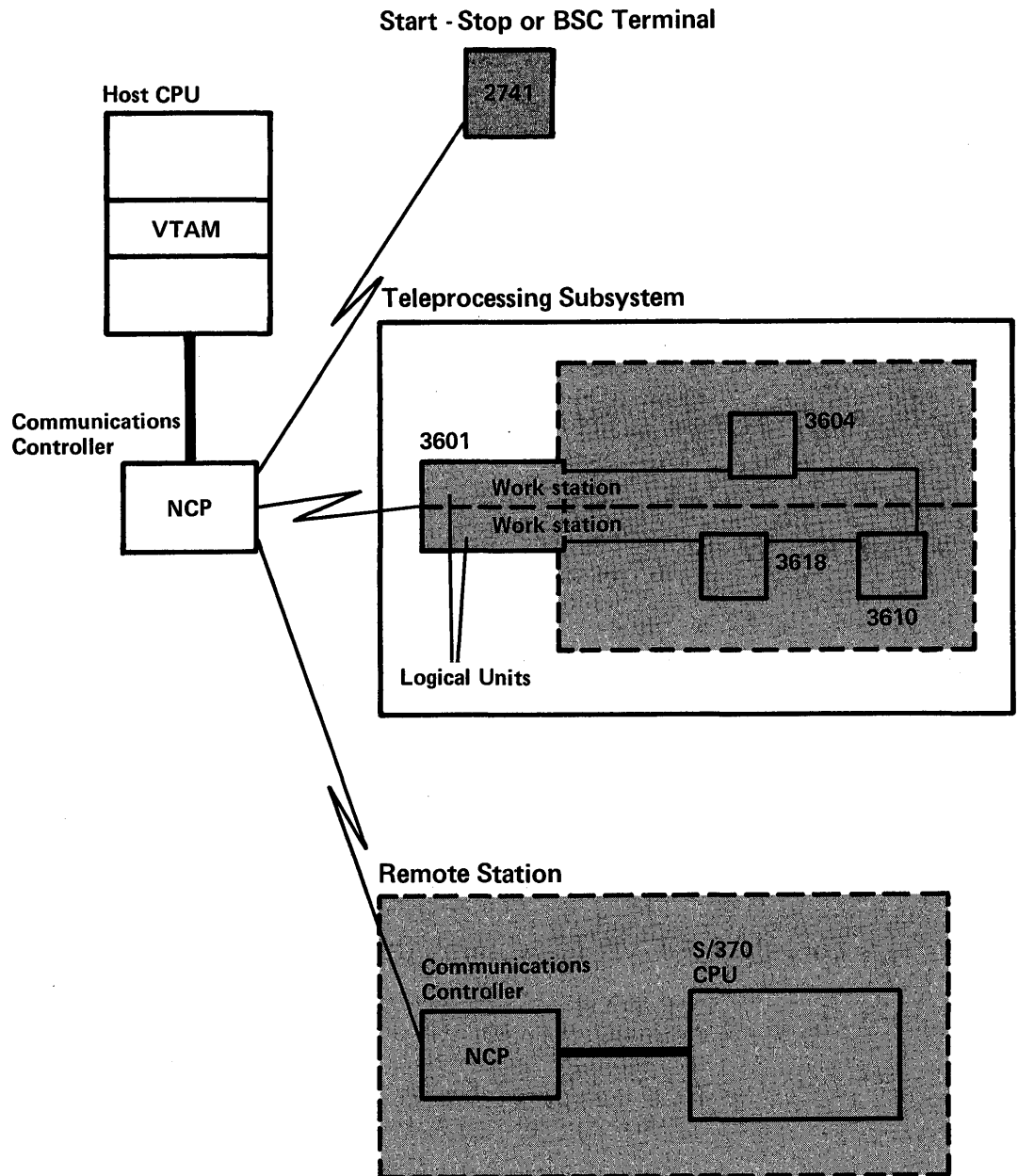
The type of line discipline used depends upon the terminal. Appendix B is a list of remotely attached devices supported by VTAM for each line discipline.

In addition to supporting remotely attached terminals, VTAM provides telecommunication support for the local attachment of the IBM 3270 Information Display System. A locally attached 3270 is a 3270 terminal whose cluster control unit (a 3272) is attached to the host CPU directly via a data channel.

In text about VTAM, the word *terminal* has a particular meaning. A terminal is a point in the telecommunication network from which data can enter or leave the network. A VTAM terminal can therefore be a single start-stop or BSC input/output device (such as the IBM 2741 Communication Terminal), a CPU complex (such as a System/370 Processor Station operating as a *remote station*), or a *logical unit* of a *teleprocessing subsystem* (such as a work station of an IBM 3600 Finance Communication System). Figure 2-3 shows the three types of terminals in the VTAM telecommunication network.

In VTAM, a logical unit is that entity of a teleprocessing system that is treated as a terminal by VTAM. (Logical units are treated as terminals in that VTAM treats both as points in the network at which data can leave or enter the network.) The exact composition of a logical unit depends upon the subsystem being supported. Figure 2-4 specifies the teleprocessing subsystems supported by VTAM and the composition of the logical units for each.

In short, the boundary of the telecommunication network for VTAM appears as a terminal to VTAM. In some cases this boundary is also the physical limit of the network; in others, it is not. For example, as shown in Figure 2-3, a 2741 terminal is a terminal to



This figure shows four VTAM terminals: a 2741, two logical units (referred to as work stations in 3600 publications), and a remote station.

Figure 2-3. Types of VTAM Terminals

VTAM; it is also the physical end of the network. For the 2741, there is a one-for-one relationship between the VTAM terminal and the physical device. On the other hand, the relationship between the VTAM terminal and the physical device is not as precise for a teleprocessing subsystem or a remote station. In a teleprocessing subsystem, VTAM treats each logical unit as a separate terminal; for example, in the IBM 3600 Finance Communication System, each work station is defined to VTAM and is treated as a separate terminal. In a remote station, VTAM treats each CPU and its attached devices as a single terminal. See Appendix C for details on VTAM's support of a remote station as a

terminal. *Note:* As used in this publication and unless stated otherwise, *terminal* also applies to NCP components.

## Distributed Function

In understanding VTAM's management of a telecommunication system, what is important to note is that VTAM is responsible for the transfer of data between the nodes in only the VTAM portion of a telecommunication system. VTAM application programs and terminals mark the limits of the VTAM system. Control of the data flow beyond these limits is the responsibility of nodes at those limits.

In a VTAM system, control functions may be distributed among nodes; rather than concentrated in the host CPU, some control functions have been distributed to nodes such as NCPs in communications controllers, cluster control units, and logical units. VTAM's management of the system takes advantage of this distribution; for example:

- By using the NCP to perform activities such as line-scheduling and polling, much of the network control and host CPU processing can be performed simultaneously.
- By relying on processing capabilities of logical units and remote stations, the composition and function of the network beyond these nodes are transparent to VTAM.

In effect, the system under VTAM's direct control is only a subset of the total telecommunication system; the remainder of the system is controlled by nodes at the edge of the VTAM system. But, while VTAM directly controls only a subset, the access method can be used to coordinate the activities of the entire system. The remainder of this publication describes VTAM's operation within that subset. The terms *telecommunication system* and *telecommunication network* are used in the remainder of this publication to refer to only the areas controlled by VTAM, unless stated otherwise.

## Data Flow Through a VTAM Telecommunication System

Note, that even though logical units and remote stations may be terminals to VTAM, they do not necessarily mark the physical end of the telecommunication network. Figure 2-5 shows how data flows between application programs and terminals. Segment A of Figure 2-5 is a generalized data-flow diagram, valid for any of the three types of terminals. VTAM manages the flow of data:

- Between itself and the application program.
- Between the CPU (using the I/O facilities of the operating system) and the communications controller.
- Between the communications controller (using the facilities of the NCP) and the terminal.

In a telecommunication system that contains teleprocessing subsystems or remote stations, there are additional communication connections. For example, Segment B of Figure 2-5 shows the data flow for a VTAM system in which the VTAM terminal is a logical unit (in this example, a 3600 work station with two physical devices).

Note that, in Segment B, VTAM manages only part of the actual data transfer required in this system: the transfer of data between the application program and the VTAM terminal (the logical unit in this case). The logical unit is responsible for the transfer of

data between the IBM 3601 Finance Communication Controller and the physical input/output devices (3604 and 3610, in this example), which mark the end of the physical network.

Any data transfer activity in which the application program uses auxiliary storage devices is the responsibility of the application program. (The application program can use an access method such as IBM's Virtual Storage Access Method (VSAM) to access auxiliary storage of the host CPU.)

## Sharing Resources—An Introduction

Because VTAM allocates network resources, it permits parts of the network to be shared among the application programs being executed in the host CPU. Shared resources include:

- Communications controllers, cluster control units, and lines—which may be used by more than one application program. Actually, an application program is unaware of communications controllers, cluster control units, and lines; it communicates only with terminals. By sharing these items, several application programs may communicate with different terminals on the same multipoint line. Also, the terminals on a single multipoint line may communicate with any of the application programs using VTAM.
- Terminals—which may be used by more than one application program. Any one terminal may communicate with any one of the application programs that is using VTAM. However, once a terminal and an application program are connected, the terminal can communicate with only that application program until released by the program.

Figure 2-6 shows how resources can be shared in a VTAM telecommunication network. Through VTAM, TCAM application programs can share resources in the same manner as VTAM application programs. For more details on resource sharing through VTAM, see “Sharing Telecommunication Resources,” in Chapter 7.

## Overview of VTAM

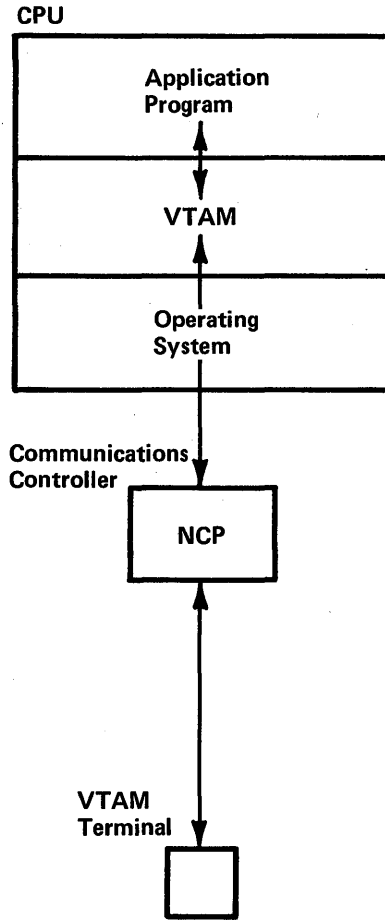
This section describes the relationship among the major elements of a VTAM system. It also introduces concepts that must be understood for proper planning and use of VTAM. These concepts are expanded later in this manual.

Teleprocessing Subsystems Supported by VTAM	Subsystem Components Treated by VTAM as Logical Units
IBM 3600 Finance Communication System	Work Station
IBM 3650 Retail Store System	3650 Application Program in the 3651 Controller
IBM 3660 Supermarket System	3660 Application Program in the 3651 Controller
IBM 3790 Communication System	3790 Application Program in the 3791 Controller

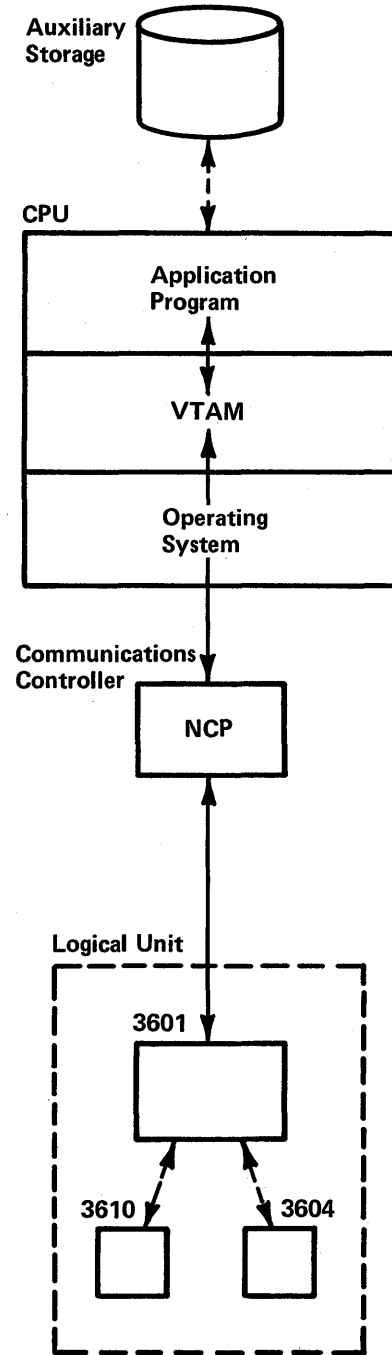
Figure 2-4. Correlation Between VTAM Logic Units and Teleprocessing Subsystems



**SEGMENT A**  
(General Flow of Data)



**SEGMENT B**  
(Flow of Data to a Logical Unit)



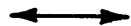
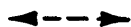
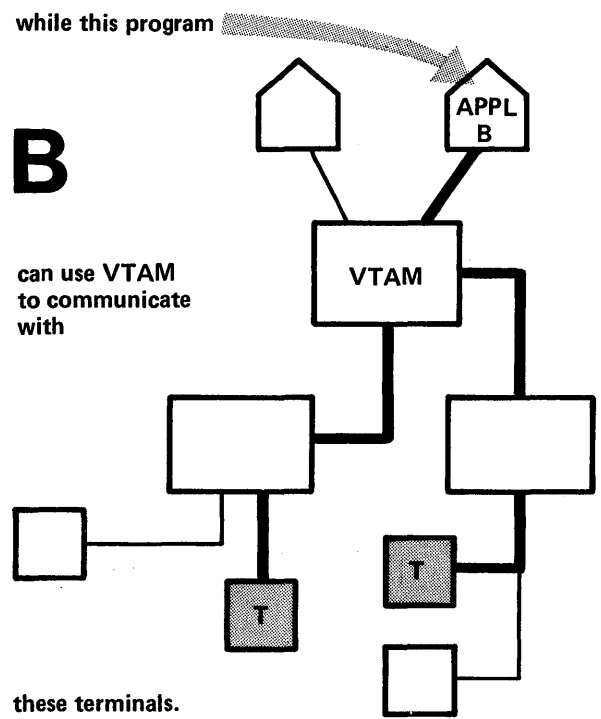
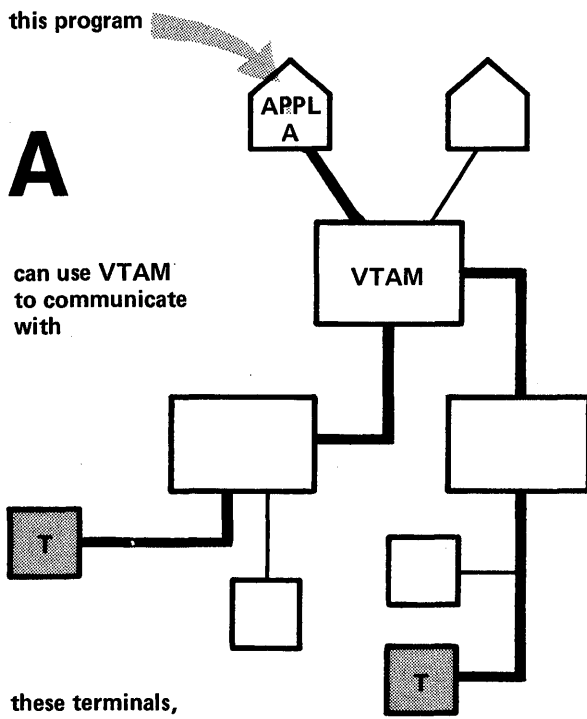
 Flow of data under VTAM's control  
 Flow of data outside of VTAM's control

Figure 2-5. Data Flow through a VTAM Telecommunication System

For example . . .



Later . . .

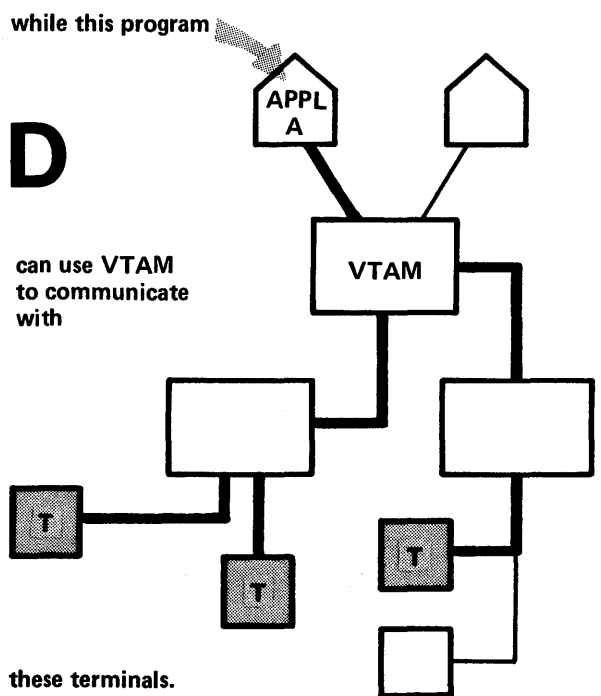
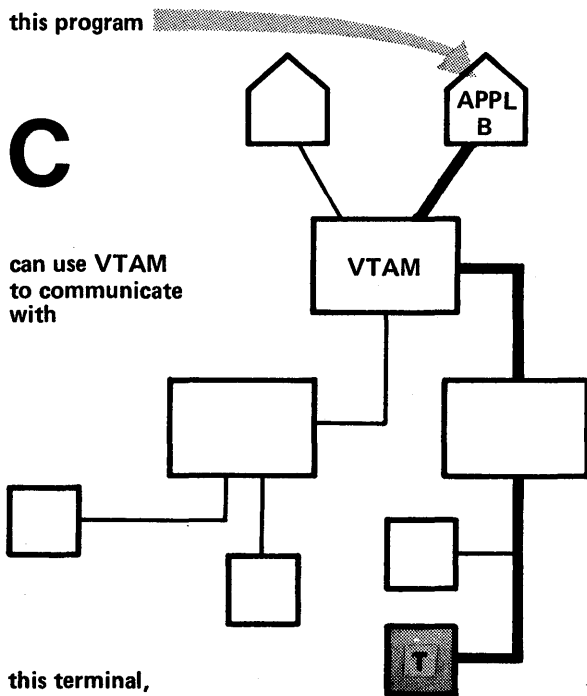


Figure 2-6. Sharing Resources

## *Four Views of a VTAM System*

To understand VTAM fully, a user should know what the system looks like from four viewpoints. These viewpoints are shown in Figure 2-7, which depicts the system: (a) as seen in terms of its physical components, (b) as seen by the operating system, (c) as seen by VTAM, and (d) as seen by application programs.

Section A of Figure 2-7 shows a possible physical configuration of the system. A VTAM system includes a central processing unit (referred to as the host CPU) with a system console (labeled S1). The system console is used to enter VTAM operator commands (called network-operator commands) to control the telecommunication system. Also attached to the host CPU are auxiliary storage devices that contain data sets used by VTAM.

The network shown in Section A includes two locally attached 3270s (labeled L1 and L2) and one locally attached communications controller. Attached to the local communications controller are three terminals (labeled T1, T2, and T3) and a remote communications controller. T1 is attached via a point-to-point line. T2 and T3 are logical units (work stations for the 3600 system). T2 includes two 3600 devices; T3 includes one 3600 device. Attached to the remote communications controller are three terminals (labeled T4, T5, and T6). T4 is attached via a point-to-point line; T5 and T6 are attached via the same multipoint line.

A telecommunication system has a definite physical configuration, but it is defined to, and used by, the operating system, VTAM, and application programs—each differently. Section B, C, and D of Figure 2-7 depict these differences.

Section B of Figure 2-7 depicts the telecommunication system as viewed by the operating system.

Support is generated in the operating system for only the system console, the auxiliary storage devices, and the locally attached devices of the telecommunication network (the locally attached 3270s and the locally attached communications controller). Support of remotely attached devices need not be generated at system generation if these attachments are to be used only through VTAM. Such support is generated within VTAM, as explained later.

The number of auxiliary storage devices used by VTAM depends upon data requirements that, in turn, are influenced by factors such as the size and complexity of the telecommunication system. In general, data used by or generated by VTAM falls into one of three categories. As shown in Section B of Figure 2-7, these categories are as follows:

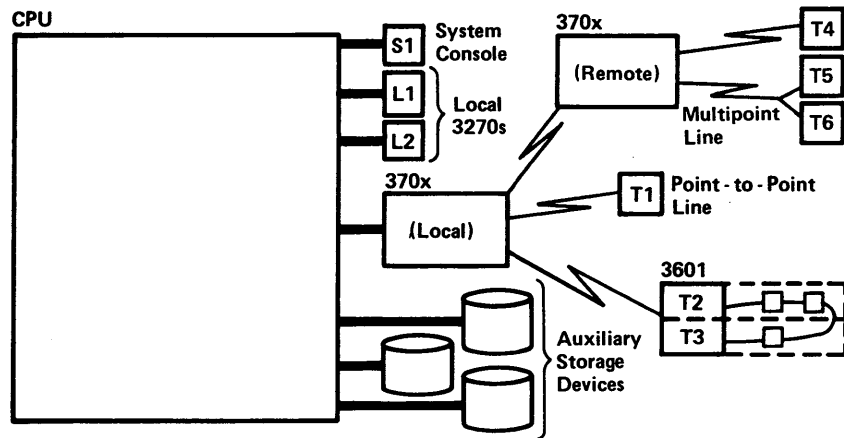
- *VTAM libraries* which contain VTAM load modules, descriptions of the telecommunication system, and operational specifications of the installation.
- *NCP libraries*, which contain NCP load modules and dump records.
- *RAS libraries*, which contain records to assist in error recording and maintenance of the VTAM system. RAS stands for reliability, availability, and serviceability.

VTAM and NCP libraries include VTAM, NCP, and operating system data sets; most of the library requirements for RAS involve operating system data sets. The composition and organization of these libraries depend upon the operating system under which VTAM is executing. (See Chapter 7 for details on operating system requirements and on data set requirements for VTAM.)

Section C of Figure 2-7 depicts the telecommunication system as viewed by VTAM.

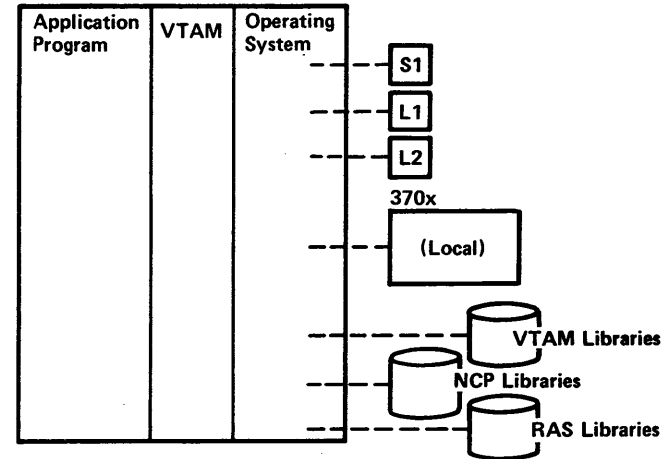
Figure 2-7. Four Views of a VTAM Telecommunication System

**A PHYSICAL CONFIGURATION OF A TELECOMMUNICATION SYSTEM**

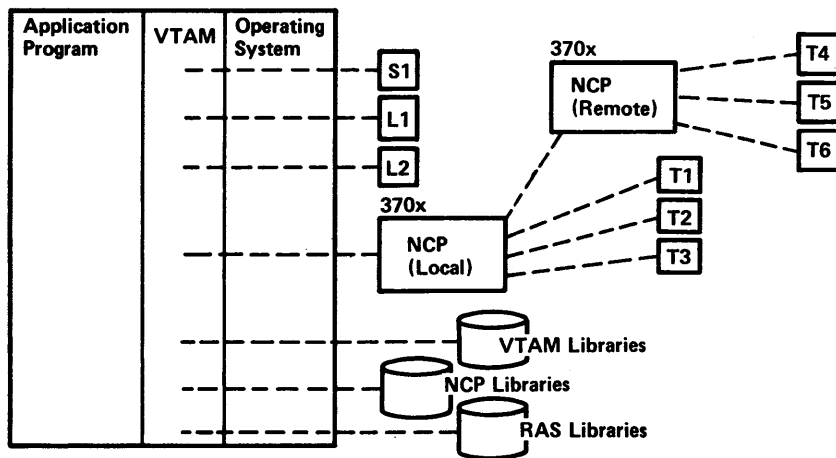


Note: T2 and T3 are 3600 Work Stations

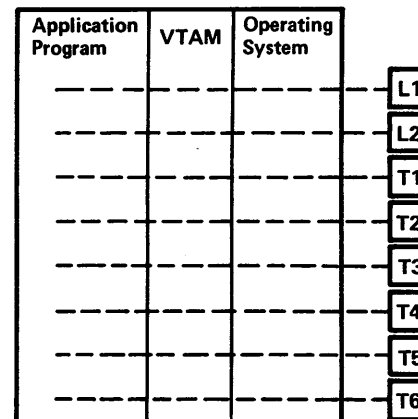
**B THE TELECOMMUNICATION SYSTEM VIEWED BY THE OPERATING SYSTEM**



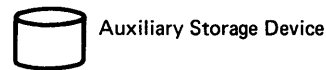
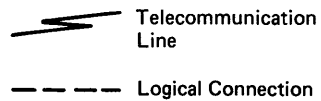
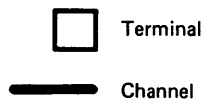
**C THE TELECOMMUNICATION SYSTEM VIEWED BY VTAM**



**D THE TELECOMMUNICATION SYSTEM VIEWED BY APPLICATION PROGRAMS**



Legend:



370x = 3704 or 3705 Communications Controller

The host CPU must contain the operating system (DOS/VS, OS/VS1, or OS/VS2), VTAM, and one or more application programs. To VTAM, an active application program is an open (that is, an initialized) access-method control block (ACB). As shown in Section B, all locally attached devices are initially "owned" by the operating system, but (as indicated in section C) when VTAM is started and begins activating parts of the telecommunication network, VTAM acquires the use of these devices. Some of the devices are allocated explicitly to VTAM, some are allocated implicitly, and others are used by VTAM but remain allocated to the operating system.

Devices that are explicitly allocated to VTAM include the locally attached 3270s and the locally attached communications controller. These devices are explicitly allocated because support for these devices is generated in the operating system, and this support is extended to VTAM. When one of these devices is allocated to VTAM, VTAM becomes its sole "owner"; the only access to the device is through VTAM. Locally attached devices not allocated to VTAM remain available for allocation by the operating system to other, non-VTAM users.

The remote attachments (terminals T1 through T6, the remote communications controller, and the 3601 controller in Figure 2-7, Section C) are implicitly allocated to VTAM. They are implicitly allocated because the only access to these devices is through the local communications controller, and VTAM controls access to this controller. *Note:* In the case of an NCP with PEP, VTAM only controls the access to that part of the remote network serviced by the network control mode of the NCP.

Also allocated to VTAM are the data sets on the auxiliary storage devices. These data sets contain the VTAM libraries, the NCP libraries, and some of the RAS libraries. RAS libraries not allocated directly to VTAM are used by VTAM through the RAS facilities of the operating system.

The system console is used by VTAM but not actually allocated to VTAM. Communication is established between VTAM and the network operator through this console. The network operator enters VTAM commands through this console, and VTAM transmits messages to the network operator at this console.

As noted previously, a primary purpose of VTAM is to provide the communication link between application programs and terminals. VTAM uses the operator services of the operating system to communicate with the network operator at the system console; it uses the operating system's data management services and, in some cases, its RAS facilities, to gain access to the libraries on the auxiliary storage devices; but communication with terminals in the telecommunication network and with application programs in the host CPU are handled directly by VTAM.

Section D of Figure 2-7 shows the telecommunication system as viewed by the application program. This view results from VTAM's ownership of all nodes in the network and from the way VTAM allocates the use of these nodes. VTAM connects application programs to only terminals; the other, intermediate, nodes are allocated only for the time needed to satisfy a specific transmission request.

As shown in Section D of the figure, application programs connect with terminals and need not be concerned with intermediate connections such as channels, communications controllers (and NCPs), and telecommunication lines. Application programs are also not directly concerned with the system console used by the network operator or with the VTAM, the NCP, or the RAS libraries.

## ***VTAM in Operation***

Activities that pertain to VTAM include defining the telecommunication system, controlling the activities of VTAM from the network-operator's console, and executing application programs that use VTAM for data transmission. Operating in conjunction with each other, these activities provide a functioning telecommunication system.

Defining the system to VTAM involves identifying and describing all of the nodes in the system. These node descriptions are collected and filed as members in OS/VS, or books in DOS/VS, in a VTAM library. Activating a node in VTAM includes using the description to establish a definition of, and control information for, the node. VTAM uses this information to control the allocation of the node.

When VTAM is started, its initiation function establishes the telecommunication system according to the specifications of the installation.

Once VTAM has been started, active application programs can connect with active terminals in the telecommunication network. As long as a terminal is connected to an application program, it can communicate with that application program.

To form a connection with a terminal, an application program must first identify itself to VTAM. Once identified, the application program can request connection to a specific terminal (or list of terminals). An application program can either acquire or accept a connection. When it acquires a connection, the initiative for the connection originates in the application program; when it accepts a connection, the connection is initiated via a logon. When a logon is requested, the terminal is "queued" to the application program; that is, the connection process is initiated but not completed. The application program must accept the terminal to complete the connection. Connection is made to the terminal, not the line. When the connection request is completed, the application program is able to transmit data (via input/output requests) to the terminal.

In transmitting data to a terminal, the data is moved from the application program's output data areas to VTAM buffers. VTAM then transmits the data to the terminal (via the NCP for remotely attached devices).

Input from the terminals travels the same (but reverse) route. The transmission moves from the terminal to VTAM (via the NCP for remotely attached devices). VTAM then moves the information to the application program input areas.

When an application program no longer needs a terminal, it can disconnect from the terminal. VTAM can then reallocate the terminal to another application program.

When the telecommunication system is to be closed down, VTAM's termination function enables the installation to orderly terminate VTAM processing and to cease telecommunication activity.

From the time that VTAM is started to the time it is terminated, the VTAM network-operator facilities enable the installation to control and monitor the telecommunication system. Most modifications to the network can be made dynamically, without having to terminate VTAM.

The steps involved in installing an operating telecommunication system with VTAM are discussed in "Installing a VTAM System," below.

## ***Installing a VTAM System***

To install a VTAM system, it must be created, procedures should be defined for using it, the active system must be controlled, and application programs must be designed and coded for the host CPU. These steps are discussed below.

## **Creating the System**

A VTAM telecommunication system is created through a process called VTAM definition, which includes generating VTAM, defining nodes in the telecommunication system, and tailoring VTAM.

Generating VTAM is part of the system generation process of the operating system. Defining nodes to VTAM is a separate process of identifying and describing them and then filing these definitions in a VTAM library. Tailoring VTAM includes coding exit-routines that perform functions such as checking the validity of connection requests between application programs and terminals, collecting information, and structuring VTAM's logon facility to the installation's specifications.

See Chapter 3 for detailed information on using VTAM to create a telecommunication system. Chapter 3 also contains more detailed information on nodes, VTAM buffering, and logons.

## **Controlling the Network**

VTAM enables a network operator to dynamically control the telecommunication network. The network operator can start and stop VTAM, monitor the activity of the telecommunication system, activate and deactivate nodes, and start and stop specified VTAM facilities. To perform these functions, the network operator is provided with a set of VTAM commands. Working within the confines of the network created through VTAM definition, the operator uses these commands to monitor and dynamically configure the network.

See Chapter 4 for detailed information on the responsibilities and actions of the network operator and for a description of the VTAM network operator facilities. Chapter 4 also contains detailed information on activation of nodes.

## **Designing VTAM Application Programs**

VTAM enables application programs to request connection with specific terminals and to request the transfer of data between the applications and their connected terminals. VTAM also provides facilities for application programs to process requests synchronously or asynchronously. Other facilities include exits that are scheduled upon the completion of specified events and an extensive error notification scheme.

Chapter 5 introduces the VTAM facilities available to the application program, and it provides suggestions on designing VTAM application programs. It also provides more detail on VTAM concepts pertinent to application programs—such as connection, communication, and synchronous and asynchronous processing.

## **Establishing Procedures for Using the System**

Once a VTAM telecommunication system has been started, it is available to application programs, terminal operators, and the network operator. To ensure that the telecommunication system is used effectively and efficiently, the installation needs to establish procedures to be followed by the users of the system and to institute controls that monitor these procedures.

Procedures should be established for the network operator for starting, stopping, and manipulating the VTAM system. The network operator needs to know how and when to activate and deactivate nodes and specific VTAM functions. The network operator also must know what to do when error conditions are encountered and what action to take to avoid unnecessary down-time. (These actions might include responding to error messages, collecting status information, or correcting the problem.)

The application programmer needs to know the conventions to be followed when connecting with VTAM and with terminals. Procedures should also be established for the interaction between the application program and the rest of the system. Such procedures might encompass passing terminal connections between application programs and reacting to system closedown.

The terminal operator may need to know how to log onto and log off from application programs.

Controls should be established to ensure that only authorized users can actually gain access to VTAM resources. VTAM facilities can be used to control connection to VTAM and between application programs and terminals. Facilities are also available to restrict the use of certain VTAM functions to only authorized users and to protect confidential data.

Chapter 6 discusses the RAS capabilities of VTAM. Chapter 7 describes various VTAM planning considerations. The exact procedures defined by an installation depends upon the needs and requirements of the installation itself coupled with the functions and facilities of VTAM as addressed in this publication.





# CHAPTER 3. CREATING A TELECOMMUNICATION SYSTEM WITH VTAM

This chapter describes how a VTAM telecommunication system is created and tailored to the installation's requirements. This chapter also contains a detailed description of nodes, VTAM buffering, and logons.

## VTAM Definition

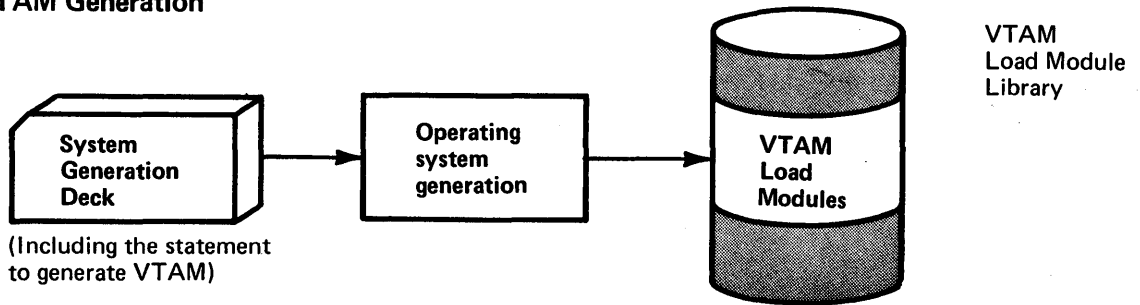
Before a telecommunication network can be used by VTAM, it must be defined to VTAM. The process of defining the network to VTAM is called *VTAM definition*. VTAM definition includes describing the physical configuration of the network, selecting the functions to be available in the telecommunication system, and otherwise tailoring the system to the installation's requirements. VTAM definition is the process of:

- A. Generating VTAM.
- B. Defining the network.
- C. Tailoring a VTAM system.

Figure 3-1 illustrates the three steps involved in defining a VTAM system. Steps B and C in Figure 3-1 can be repeated as often as needed without requiring a repetition of step A.

The remainder of this section describes in detail each of the steps shown in Figure 3-1.

### A VTAM Generation



### B Network Definition

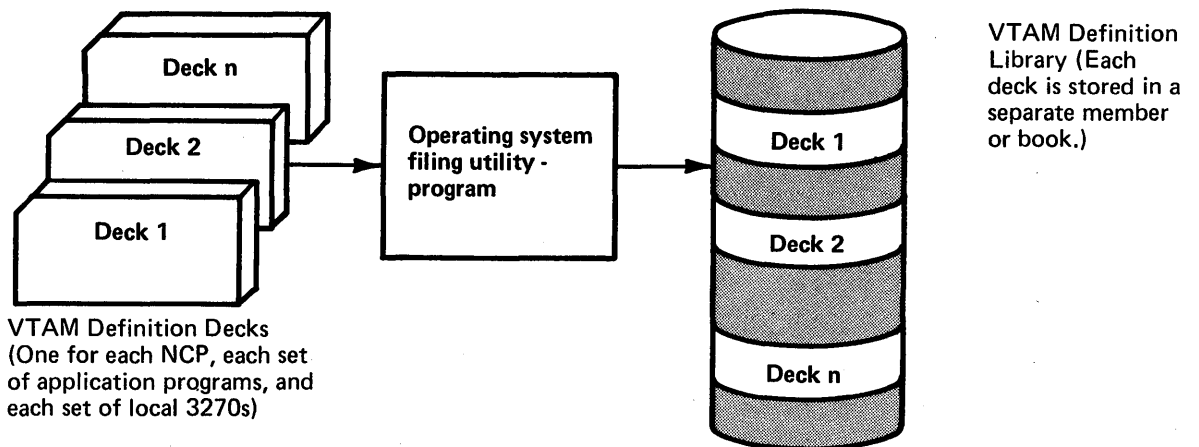


Figure 3-1 (Part 1 of 2). Defining a VTAM System

## C Tailoring VTAM

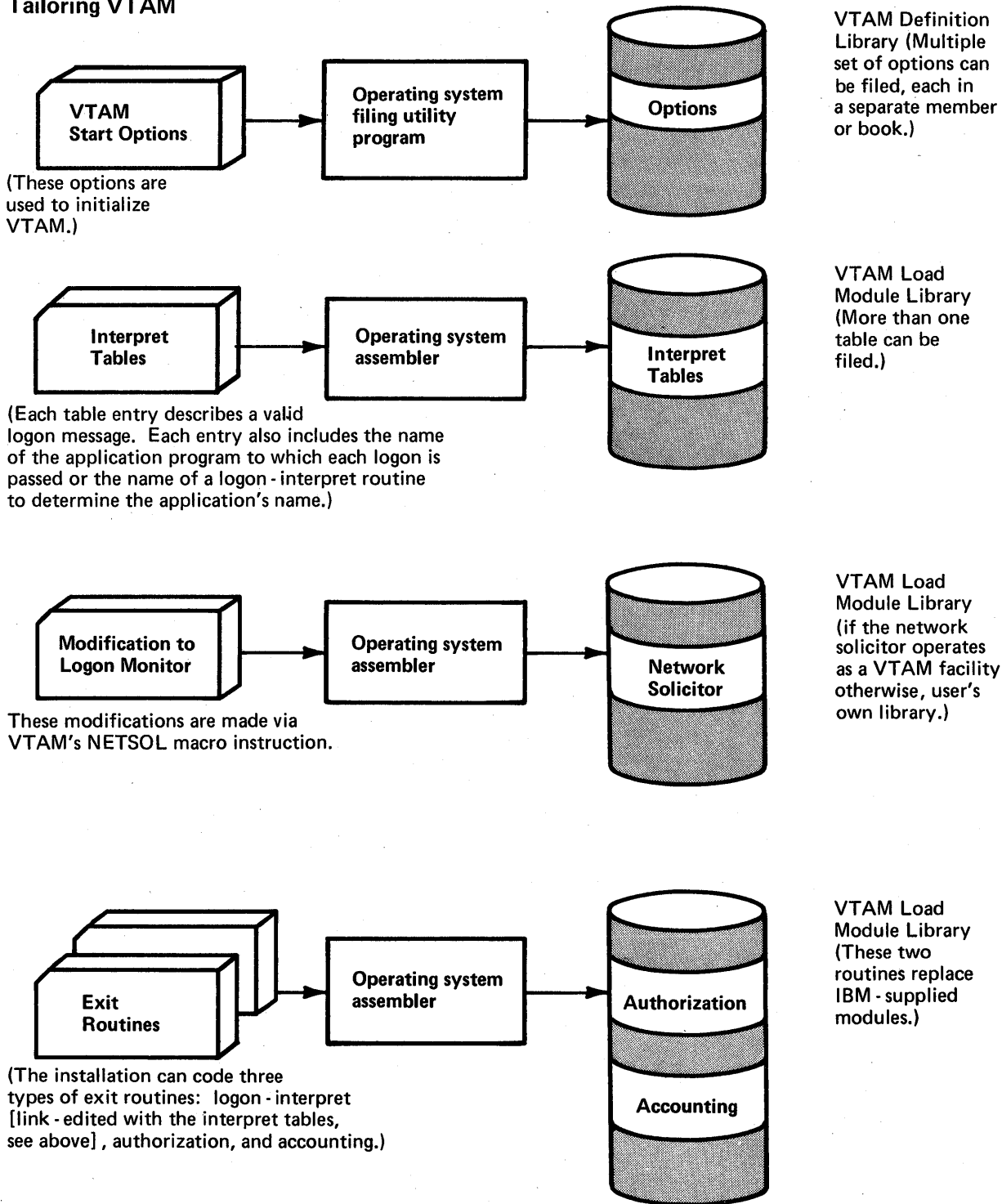


Figure 3-1 (Part 2 of 2). Defining a VTAM System

## ***Generating VTAM***

During system generation, VTAM modules are generated and included in the operating system. To generate the VTAM modules and the required support in the operating system, the following are specified in the input stream for the first stage of system generation:

- VTAM is specified as a parameter of the TP operand of the SUPVR macro instruction for DOS/VS, or VTAM is specified as a parameter of the ACSMETH operand of the DATAMGT macro instruction for OS/VS.
- Control-unit and device statements are specified *only* for locally attached devices to be used by VTAM; that is, only for locally attached 3270s and locally attached communications controllers.

*Note:* Remotely attached devices are not specified during system generation. These devices are specified and described during network definition. Remember that network definition can be done online, without disrupting other jobs in the operating system. (See “Defining the Network,” below, for a description of network definition.)

Additional operating-system support for VTAM can be included at system generation. See “Operating System Requirements,” in Chapter 7, for details on operating-system support.

## ***Defining the Network***

Network definition is the process of describing the network configuration to VTAM. These descriptions are coded in VTAM definition statements; the coded definitions are then filed in the *VTAM definition library*. This library is SYSSLB for DOS/VS and SYS1.VTAMLST for OS/VS. See “VTAM Data Sets,” in Chapter 7, for a detailed description of this library.

As part of the network definition, the four types of *major nodes* in the telecommunication system are defined to VTAM:

- NCPs for locally attached communications controllers, including their attached terminals.
- NCPs for remotely attached communications controllers, including their attached terminals.
- Sets of locally attached 3270s.
- Sets of application programs that use VTAM.

The definition of each major node is filed as a separate member (specifically, a member in OS/VS or a book in DOS/VS) of the VTAM definition library. When VTAM activates a major node, it uses the filed definition of that node as a description of the node’s configuration. The defining of a major node can occur anytime following system generation but prior to the first use of that node by VTAM. See “VTAM Node Structure,” later in this chapter, for a description of nodes and a detailed definition of major node.

### **Defining NCPs for Communications Controllers**

At least one group of definition statements must be provided to VTAM for each communications controller in the VTAM network. To aid in defining the remote network for a communications controller, the same deck of macro instructions used to generate an NCP can be used as the node definition by VTAM. Using this deck both for NCP generation and for VTAM network definition requires additions to the NCP generation macro instructions. These additions include statements and parameters that are used only by VTAM. See Chapter 7 for VTAM requirements for an NCP.

An NCP configuration is defined by filing the NCP generation deck as a unique member (member in OS/VS, book in DOS/VS) of the VTAM definition library. The member name

is thereafter used when addressing the NCP through VTAM. Each NCP defined to VTAM is called a major node. See "VTAM Node Structure," later in this chapter, for a detailed definition of major node.

If an NCP is modified (regenerated), it must be redefined to VTAM. It is redefined by refiling the altered or new NCP generation deck. Thus, remote network configurations can be modified without requiring a new or partial system generation of the operating system.

Figure 3-2 depicts the steps for generating NCP support in a VTAM telecommunication system. The steps are as follows:

1. *Planning the NCP.* Keep VTAM requirements, restrictions, and considerations in mind.
2. *Coding the NCP generation statements.* Include the parameters and definitions statements required by VTAM as well as those used to generate the NCP.
3. *Generating the NCP.* Use the statements coded in step 2. Include those parameters and definition statements that are used only by VTAM; these parameters and statements, though not used to generate the NCP, undergo an initial verification by the NCP generation process.
4. *Verifying that the generation is successful.* If the NCP is not generated, successfully, correct the generation deck and repeat step 3.
5. *Filing the generation deck.* File the deck as part of a member (member in OS/VS, book in DOS/VS) of the VTAM definition library. This deck is the same deck that was coded in step 2 and used in step 3. When VTAM activates this NCP, VTAM extracts the information it needs from the filed definition and from the generated NCP itself.

Using the same deck to generate an NCP and to define it to VTAM ensures that the generated NCP and its VTAM definition agree. The deck is filed on the definition library using a utility of the operating system.

Including the VTAM-only information in the generation deck permits an initial verification of these specifications during NCP generation. Generating the NCP prior to filing the deck ensures that the NCP generated is the one that is defined. If the deck is filed first and an error is encountered in the generation process, the updated deck would have to be refiled.

**Note:** The statements used to generate the NCP can be referred to as macro instructions because they are assembled and generate communications controller instructions. As used by VTAM, these same statements are not assembled and do not generate instructions. Therefore, in this publication, they are referred to as statements, not macro instructions.

### Defining Locally Attached 3270s

Each locally attached 3270 terminal must be defined to VTAM, either individually or as part of a logical set of locally attached 3270s. Definitions of locally attached 3270s are provided in VTAM's LOCAL definition statements. A LOCAL statement defines one terminal (a printer or a display unit), and each locally attached terminal in a VTAM network must be defined by at least one LOCAL statement.

A logical set of locally attached terminals can be defined by filing the LOCAL statements, one for each terminal in the set, along with an LBUILD statement as a member (member in OS/VS, book in DOS/VS) of the VTAM definition library. (The LBUILD statement identifies a major node—a term that is defined in "VTAM Definition," later in this chapter.) A terminal can be included in more than one logical set, as shown in Figure 3-3.

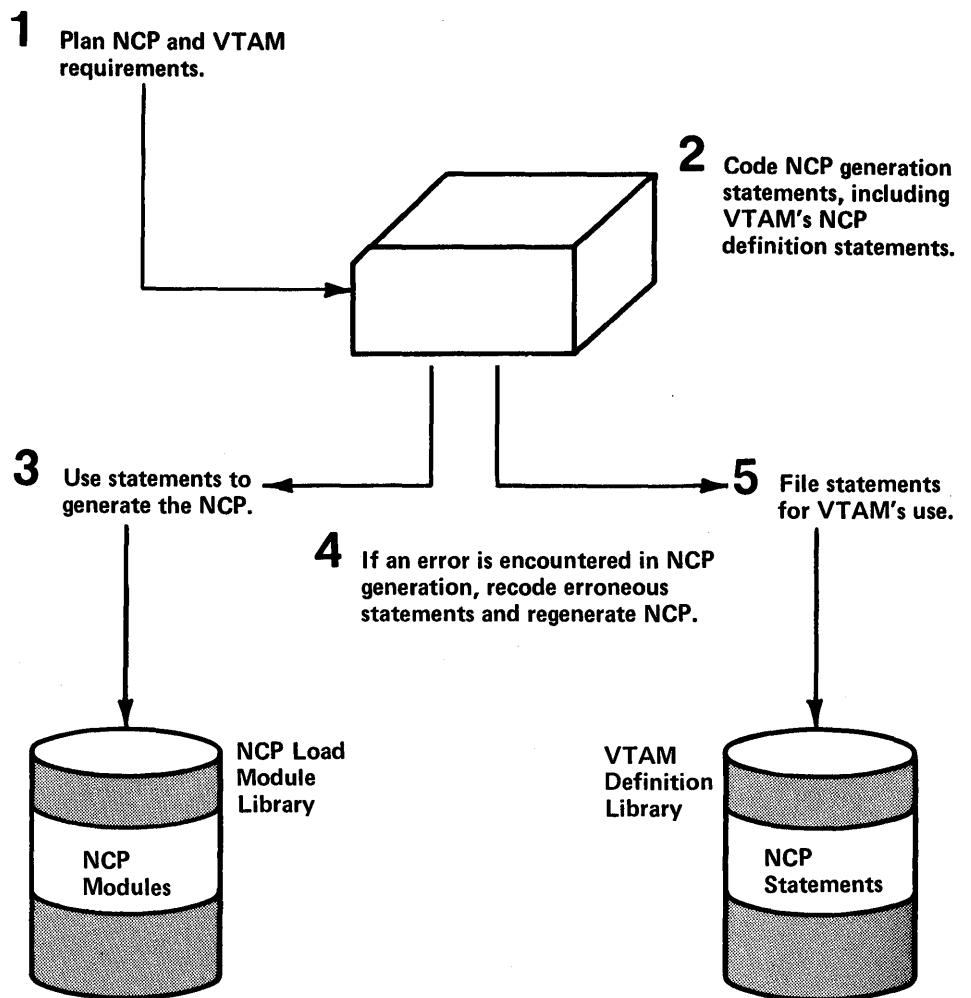
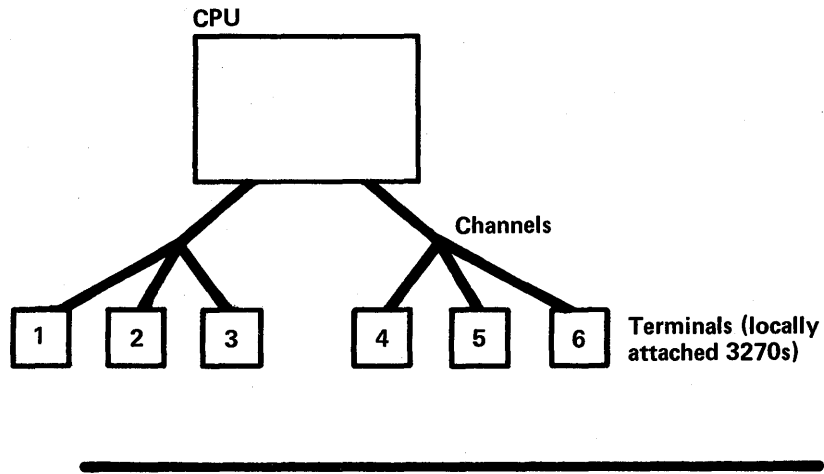


Figure 3-2. Generating NCP Support in a VTAM Telecommunication System

The following information is provided via the LOCAL statements:

- The symbolic name of the terminal.
- The channel and unit address of the terminal.
- The features that are available on the terminal.
- The name of the logon description (interpret table) to be used when analyzing logon requests for the terminal. A terminal's interpret table is also inspected whenever an INTRPRET macro instruction is issued by an application program for that terminal. (See "Tailoring a VTAM System," later in this chapter, for a description of defining logons in interpret tables. See "The VTAM Language," in Chapter 5, for a description of VTAM's INTRPRET macro instruction.)
- The name of an application program to which VTAM is to automatically transmit a logon, on behalf of the terminal, whenever the terminal is available for connection. See "Tailoring a VTAM System," later in this chapter, for a description of automatic logon.
- Whether the terminal is to be considered active or inactive when the logical set of which it is a part is activated.
- The buffer limit for the terminal (an option in OS/VS only). See "VTAM Buffering," later in this chapter, for a description of how buffer limits are established using this specification.

**PHYSICAL CONFIGURATION**



**NETWORK DEFINITION**

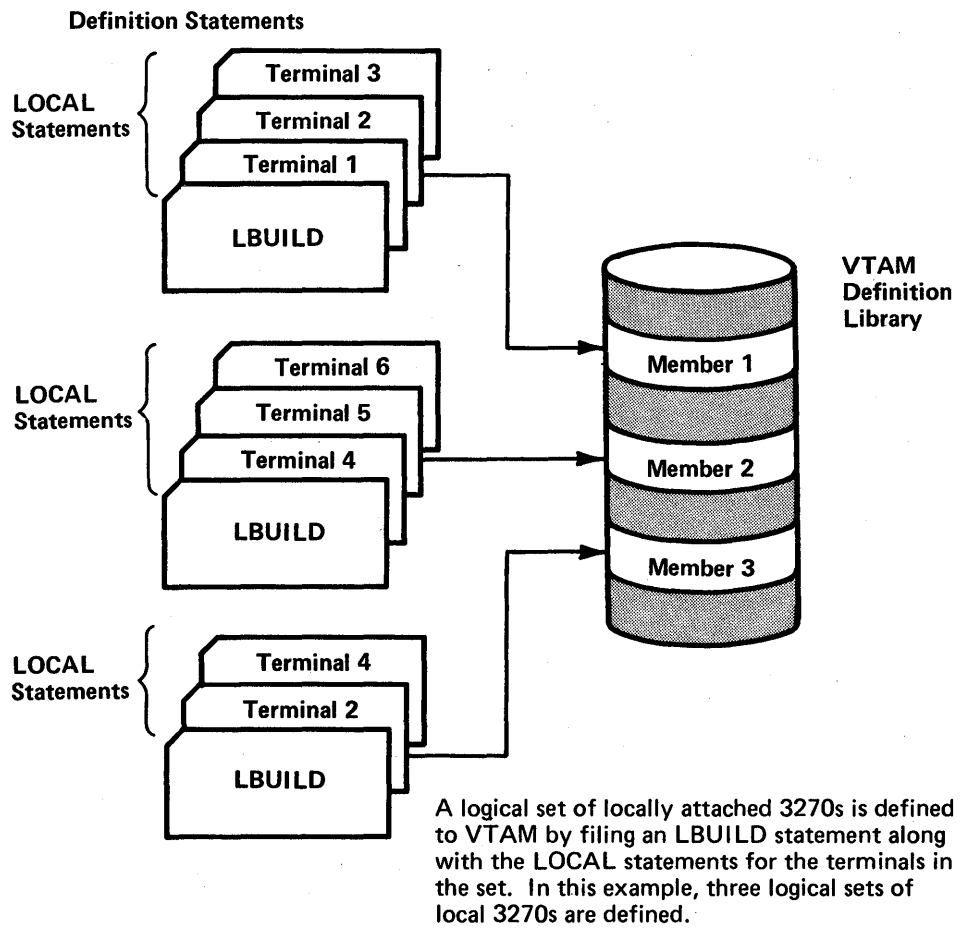


Figure 3-3. Grouping Locally Attached 3270s into Logical Sets

## Defining Application Programs

Application programs must also be defined to VTAM. They are defined with VTAM's APPL definition statement. The name specified in the APPLID field of the ACB points to a name stored in the application program that must match the name of an APPL statement. To VTAM, an application program is an open access-method control block (ACB).

APPL statements can be filed as separate members (members for OS/VS, books for DOS/VS) of the VTAM definition library, or they can be grouped in various combinations and filed as sets.

The APPL definition statement can provide the following information:

- The symbolic name to be referenced by APPLID in the ACB.
- The password to be specified by the application program when the ACB is opened.
- The buffer factor for the application program (an option for OS/VS only). See "VTAM Buffering," later in this chapter, for a description of how buffer limits are established using the buffer factor of the application program.
- The specification of those VTAM facilities that an application program using this APPL definition is allowed to use.

An application program can be authorized (via its APPL statement) to perform each of the following through VTAM:

- Request input data from start-stop or BSC terminals in blocks instead of in messages or transmissions.
- Pass connections to another application program. (That is, issue a CLSDST macro instruction with the PASS option.)
- Initiate connection requests for terminals. (That is, issue the OPNDST macro instruction with the ACQUIRE option or issue the SIMLOGON macro instruction.)

An APPL statement must be defined for each unique ACB, although the same APPL statement can be named by only one open ACB at any one time. Like locally attached 3270s, an application program can be defined as part of more than one logical set of application programs; that is, the same APPL statement can be filed in more than one member of the VTAM definition library. (See Chapter 5 for descriptions of the ACB and the OPNDST, CLSDST, and the SIMLOGON macro instructions.)

A logical set of application programs filed in the same member of the VTAM definition library is called a major node. See "VTAM Node Structure," later in this chapter, for a definition of major node.

## Tailoring a VTAM System

The installation can tailor VTAM by modifying IBM-supplied facilities. Tailoring activities include:

- *Defining VTAM start options.* The installation can specify selected functions that are to be activated when VTAM is started.
- *Defining logons.* The installation can define logon messages and can indicate the application programs to which each logon request is to be routed.
- *Modifying the network solicitor.* The installation can use the IBM-supplied facility, called the network solicitor, or can create its own facility for processing logons from start-stop and BSC terminals (including locally attached 3270s). A default network solicitor is generated when VTAM is generated, during system generation. Options are also available to tailor or replace the network solicitor.



- *Coding and including installation exit-routines.* The installation can code three types of exit-routines to control or collect statistics on network operation.

Each of these tailoring activities is discussed below.

## Defining VTAM Start Options

When VTAM is started, options can be used to define the initial VTAM network and to select optional VTAM facilities. These start options can be specified by the operator as part of the START command (in OS/VS only) or as a response to the prompting of VTAM. The options can also be predefined and filed on the VTAM definition library. (See "Starting VTAM," in Chapter 4, for information on specifying options via the system operator's console.)

The VTAM start options are used to tailor VTAM to the installation's needs each time the telecommunication system is started. Predefining start options relieves the network operator of handling this activity. In addition, more than one version of the start options can be predefined, each version specifying a different VTAM configuration. With different sets of predefined options, the installation can initialize a particular VTAM system merely by selecting the appropriate set of options.

The predefined start options are stored under the member name ATCSTRxx (where xx and a data set must be filed under this name even if it does not contain any options. Other ATCSTRxx data sets can be created, but the specific data set required is specified by the installation at VTAM start-time.

The following start information can be supplied in the ATCSTRxx data set:

- Whether the VTAM logon monitor facility for start-stop and BSC terminals (referred to as the network solicitor) is to be started when VTAM is started.
- Which nodes are to be traced by the VTAM trace facility.
- Which major nodes are to be activated during start processing (VTAM initialization).
- The size of VTAM storage pools.
- The maximum number of NCP and local 3270 major nodes that will be active at any one time.
- Whether prompting messages should be sent to the network operator to obtain additional start information.

If the network solicitor is activated, it begins monitoring the active terminals assigned to it for logons. See "The Network Solicitor," later in this chapter, for a description of the network solicitor.

The trace facility is activated for specified terminals, and the trace continues for as long as the terminals are active or until the network operator stops the trace for the terminals. See "Starting and Stopping VTAM Facilities," in Chapter 4, for information on stopping the VTAM traces. See "Serviceability Aids," in Chapter 6, for a description of VTAM's trace facility.

The names of major nodes to be activated when VTAM is started are stored as a member of the VTAM definition library. These names must be stored under the member name ATCCONxx (where xx is a two-character identification created by the installation). ATCCON00 is a default member name; other ATCCONxx member names must be specified by the installation during start processing. All ATCCONxx members must be created and filed by the installation. See "VTAM Node Structure," later in this chapter, for a definition of a major node.

To activate major nodes during VTAM start processing, an ATCCONxx member is specified as a start option. This member then contains a list of the major nodes to be activated. This option enables the installation to specify the initial telecommunication configuration.

Storage pools are used by VTAM to allocate space for control blocks, buffers, and channel programs. Pools are established in both fixed and pageable storage. See "VTAM Buffering," later in this chapter, for more information on specifying storage pools and on VTAM's use of these pools.

If the network operator is to be prompted, VTAM transmits messages to the system operator's console requesting that start options be entered from the console. The network operator is prompted only (1) if prompting is requested in the ATCSTROO data set and (for OS/VS) no start parameters are entered with the START command or (2) if an error is encountered by VTAM during VTAM initialization. See "Starting VTAM," Chapter 4, for information on the role of the network operator in starting VTAM.

An installation can prepare and file multiple start-parameter members and multiple node-name members. Various combinations of these parameters can thus be used during start-time to initialize a tailored VTAM system.

## Defining Logons

This section describes how automatic and terminal-initiated logon procedures and messages can be set up. It also provides suggestions on how to define logoff procedures. The reader can refer to "Logon Types," later in the chapter, for detailed explanations of the types of logons recognized by VTAM.

**Defining Automatic Logons:** An installation can specify that a logon request is to be automatically generated on behalf of a terminal for a selected application program whenever that terminal is available. This specification is made by coding the name of the program in the terminal's GROUP, LINE, CLUSTER, VTERM, TERMINAL, LU, or LOCAL definition statement. The application program name is the name specified for that application program on its APPL definition statement, or in the case of terminal-initiated logons from start-stop or BSC terminals (including locally attached 3270s), it is the name, NETSOL, of the logon monitor facility (network solicitor) of VTAM. Note, however, that NETSOL is never specified in an LU statement because logical units are not handled by the network solicitor.

See "Network Control Program Requirements," in Chapter 7, for descriptions of GROUP, LINE, CLUSTER, VTERM, TERMINAL, and LU definition statements. See "Defining Locally Attached 3270s," earlier in this chapter, for a definition of the LOCAL statement. See "Activating and Deactivating Nodes," in Chapter 4, for a detailed description of an active terminal.

**Defining Terminal-Initiated Logons:** Defining terminal-initiated logons for start-stop and BSC terminals (including locally attached 3270) is slightly different from defining them for logical units; thus, the definition processes are discussed separately below.

**Terminal-Initiated Logons For Start-Stop and BSC Terminals:** To enable a terminal operator to issue a logon request from a start-stop or BSC terminal (including locally attached 3270s), the steps are as follows:

1. Modify the VTAM logon monitor facility for start-stop and BSC terminals (referred to as the network solicitor). (This step is optional.)
2. Define terminal operator logon procedures and messages to VTAM.
3. Activate the network solicitor.

4. Activate the terminal.
5. Enter a logon request to be processed by the network solicitor.

Steps 1 and 2 are done as part of VTAM definition. (See "The Network Solicitor," later in this chapter, for details on step 1; step 2 is part of the process of logon definition and is discussed below.) Steps 3 and 4 are completed by the network operator, although the degree of network-operator involvement depends upon the VTAM-definition options selected. (See Chapter 4 for details on activating the network solicitor and terminals.) Step 5 is accomplished by the terminal operator.

To use VTAM's network solicitor, the installation must define:

- A. Which terminals are to be handled by VTAM's network solicitor.
- B. What is the format and content of each logon message and what is the name of each application program to be notified for each logon request.
- C. Which logon messages can be used by each terminal.

An installation can use VTAM's automatic logon capability to accomplish item A. Instead of specifying an application program name for automatic logon in a terminal's GROUP, LINE, CLUSTER, VTERM, TERMINAL, or LOCAL definition statement, the installation specifies VTAM's network solicitor. Whenever a terminal so designated is available, the network solicitor monitors it for a logon request. The installation can also use the network-operator logon option to temporarily assign a terminal to the network solicitor. The network operator can assign a terminal to the network solicitor by logging the terminal on the network solicitor. See "Characteristics of Logon," later in this chapter for a description of the network-operator logon.

Interpret tables define valid logon messages to VTAM and indicate which application programs are to be notified of the connection request for each valid logon message (item C). In OS/VS, VTAM also provides a standard logon message that does not use an interpret table. See "Specifying Interpret Tables," below, for details on setting up interpret tables. See "Establishing Standard Logons," below, for details on the standard logon message.

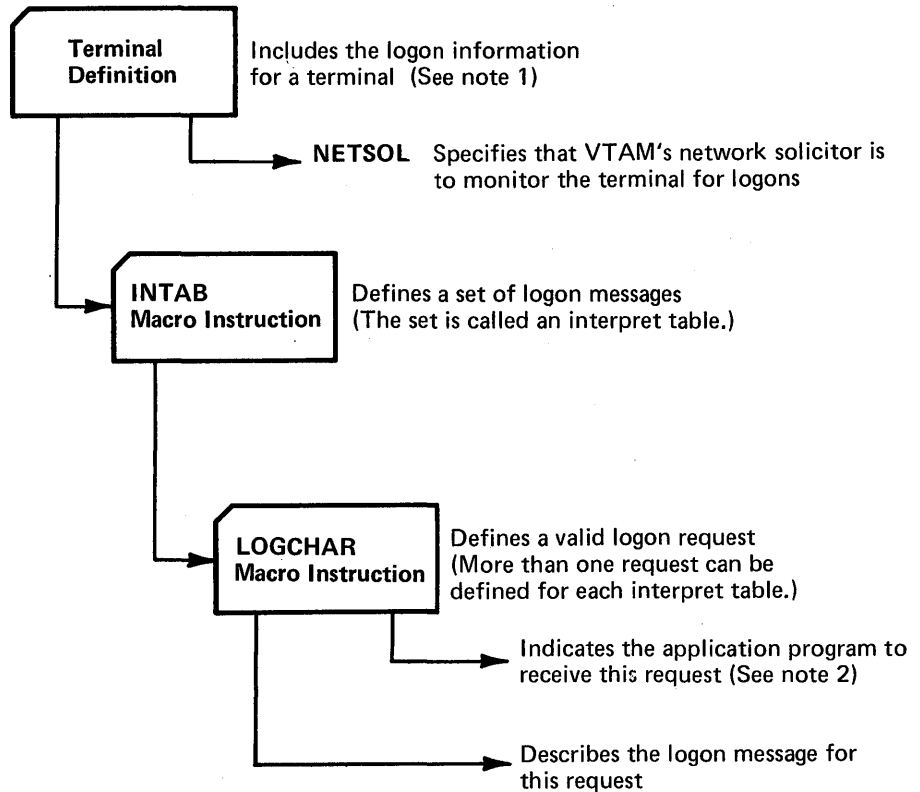
Item C is accomplished by naming in a terminal's GROUP, LINE, CLUSTER, VTERM, TERMINAL, or LOCAL definition statement the interpret table to be used to validate logon messages from this terminal. Figure 3-4 shows how control information for processing terminal-initiated logons are defined to VTAM for start-stop and BSC terminals.

*Terminal-Initiated Logons for Logical Units:* The steps involved in enabling and executing a terminal-initiated logon from a logical unit are as follows:

1. Define logon procedures and messages to VTAM.
2. Activate the terminal.
3. Enter a logon request to be processed by VTAM.

Step 1 is part of VTAM definition and is described below. Step 2 is completed by the network operator, although the degree of network-operator involvement depends upon the VTAM-definition options selected. (See Chapter 4 for details on activating terminals at start time and dynamically while VTAM is executing.) Step 3 is a function of the logical unit.

Terminal-initiated logons from logical units are not processed by VTAM's network solicitor; although, as noted above, logon procedures and messages must be established



*Notes:*

1. The logon information can be specified in the GROUP, LINE, CLUSTER, VTERM, TERMINAL, or LOCAL statement.
2. This parameter can point to an actual application program or to an installation - coded exit-routine (the interpret-logon routine) that determines the application program to receive the request.

Figure 3-4. Providing Control Information for Processing Logon Requests from Start-Stop and BSC Terminals

during VTAM definition. To enable VTAM to process logons from logical units, the installation must define:

- A. What is the format and content of each logon message and what is the name of each application program to be notified for each logon request.
- B. Which logon messages can be issued by each terminal.

Interpret tables define valid logon messages to VTAM and indicate which application programs are to be notified of the connection request for each logon message (item A). VTAM also provides a standard logon message that does not use an interpret table. See "Specifying Interpret Tables," below, for details on setting up interpret tables. See "Establishing Standard Logons," below, for details on the standard logon message.

Item B is accomplished by naming in a logical unit's GROUP, LINE, CLUSTER, or LU definition statement the interpret table to be used to validate logon messages from this terminal.

**Specifying Interpret Tables:** Interpret tables are constructed by using VTAM's INTAB, ENDINTAB, and LOGCHAR macro instructions. For purposes of defining logon messages, the LOGCHAR macro instruction describes a single logon message. The INTAB and ENDINTAB macro instructions specify a group of logon messages, each message defined by a LOGCHAR macro instruction. This group is called an *interpret table*. Each interpret table must be assembled and filed as a separate member (member in OS/VS or book in DOS/VS) in the VTAM load module library. The name assigned to the member or book is the name assigned (via the INTAB macro instruction) to the interpret table.

Thus, the INTAB and the ENDINTAB macro instruction are used to define a group of logon message definitions and to provide a name for that group. The LOGCHAR macro instruction describes a specific message and can be used to indicate the following:

- Whether the logon request is a character string or a program function key on a 3270.
- If a program key, which key.
- If a character string, what character string. The characters specified in the LOGCHAR macro instruction are only the characters to be checked by VTAM at the beginning of the message. The actual message entered from the terminal can contain additional data to be used, for example, for password protection or accounting by the application program. This additional data must not be specified in the LOGCHAR macro instruction.
- The name of the application program to receive this logon request.

For each message, the installation can specify in the LOGCHAR macro instruction either the name of an application program or the name of a routine (a logon-interpret routine) that is to determine the appropriate application program. All logon-interpret routines specified in the same interpret table must be link-edited with that interpret table.

The contents of an interpret table are accessed via VTAM's application-program INTRPRET macro instruction.

The network solicitor uses the INTRPRET macro instruction to validate logon requests. (The macro instruction can be used similarly by application programs.) The following description of the network solicitor's use of the INTRPRET macro instruction and of the interpret tables is provided as an explanation of a possible use of VTAM's interpret function.

The network solicitor invokes INTRPRET while specifying a logon message received from a terminal and the name of that terminal. INTRPRET then determines if an interpret

table is specified for that terminal. (If one is not specified, INTRPRET returns to the network solicitor, with an indication that a table is not specified.)

If a table is specified, INTRPRET checks for a match between the message passed to it and one defined by a LOGCHAR macro instruction for the table. (If no match is found, INTRPRET returns to the network solicitor, with an indication that the logon message is not in the table.)

If a match is found, INTRPRET determines whether an application program or a logon-interpret routine is specified in the LOGCHAR macro instruction. If an application program is specified, INTRPRET returns the name of the program to the network solicitor.

If a logon-interpret routine is specified, INTRPRET invokes the routine. This routine is installation coded and should validate the logon request. The logon-interpret routine should specify the name of the application program to receive the logon request; otherwise, it should specify that the logon request is invalid. If valid, the output from the routine is returned to the network solicitor.

Upon entry to a logon-interpret routine, the following information is available:

- The name of the terminal requesting the logon.
- The logon message.

Output from a logon-interpret routine should be:

- An indication of whether the logon is valid.
- The name of the application program to receive the logon if it is valid.

The message given a logon-interpret routine as input is the message from the terminal, and can therefore contain more data than is specified in the associated LOGCHAR macro instruction. The routine can use this additional data to determine the application-program name. In addition, this data might also contain information such as a password which is verified by the routine.

Although the interpret tables are intended primarily for validating terminal-initiated logon messages, they are also available to application programs through VTAM's INTRPRET macro instruction. See "The VTAM Language," in Chapter 5, for additional information on this macro instruction.

***Establishing Standard Logons:*** VTAM also provides a standard logon format that can be used in place of logons defined in interpret tables. In DOS/VS, this standard format is available to only logical units. In OS/VS, this standard format is available to start-stop and BSC terminals (including locally attached 3270s), as well as to logical units.

To enter a standard logon, a start-stop or BSC terminal must be assigned (via automatic logon) to the network solicitor, and no interpret table can be specified for the terminal. If the network solicitor encounters a logon message from a terminal that does not have an interpret table specified, it assumes the message is a standard logon. A standard logon message must have the word LOGON as the first word. The name of the application program to receive the logon request must also be included in the message. Optionally, the installation can also define data to be entered as part of the logon message. The network solicitor does not check the data, if entered. This installation-defined data could therefore be used by the application program as a password, for accounting, or for other purposes.

The standard logon for a logical unit specifies the name of the application program as part of the logon request and can include installation-defined data. A standard logon can only be issued by a logical unit for which no interpret table is specified.

**Defining Logoffs:** No logoff data is defined for VTAM. Logoff is a function of the application program, and because of this, the installation may want to establish procedures to be followed by the terminal operators and by the application programmers for logging off. See "Logoffs," later in this chapter, for suggested logoff processing.

## The Network Solicitor

This section describes VTAM's logon monitor facility for start-stop and BSC terminals (referred to as the network solicitor) and describes how it can be modified.

The network solicitor monitors start-stop and BSC terminals (including locally attached 3270s) for logons and passes terminals with valid logons to the appropriate application programs. Using the network solicitor, an installation can permit terminal-initiated logons for start-stop and BSC terminals.

**Network Solicitor—A Description:** Figure 3-5 shows how the network solicitor functions. Terminals are monitored by the network solicitor if they are assigned to it (via the automatic logon specification) and only when they are active but not connected, or queued for connection, to an application program.

**Terminal operator enters logon from active start-stop or BSC terminal.**

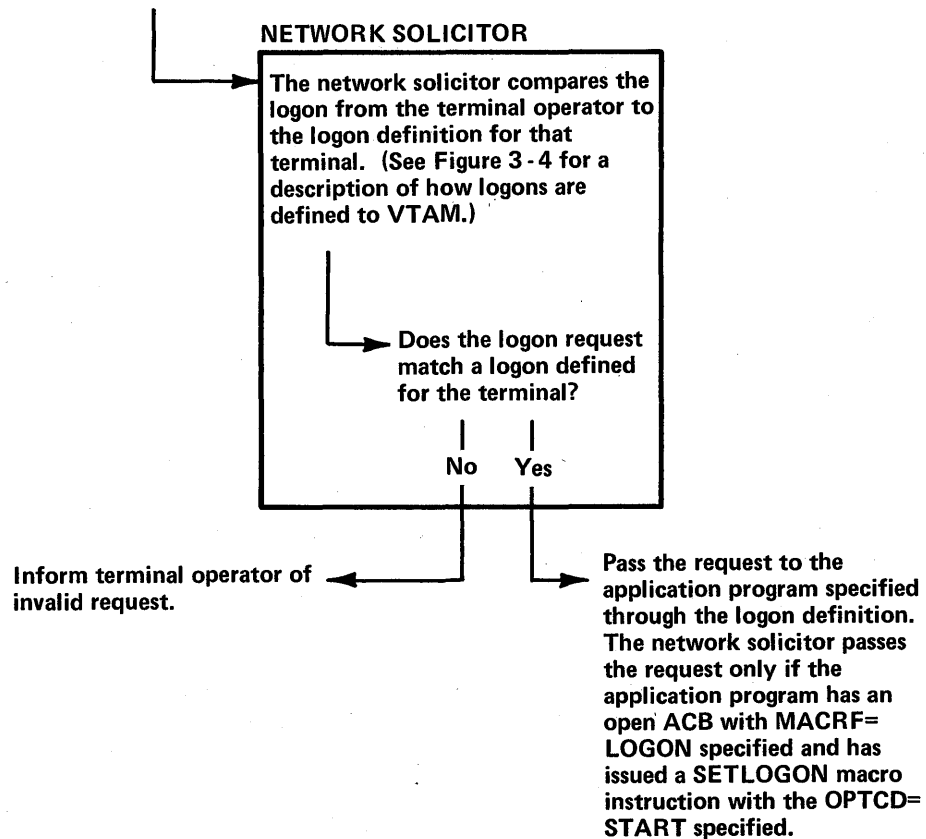


Figure 3-5. Processing a Terminal-Initiated Logon with the Network Solicitor

When a terminal being monitored by the network solicitor enters a message, the network solicitor determines whether the message is a valid logon request. The message is validated in one of two ways:

- If an interpret table is specified for the terminal, a search is made in that table for an entry corresponding to the message.
- If no interpret table is specified and the operating system is OS/VS, the message is checked for standard format.

See “Defining Logons,” earlier in this chapter, for a description of defining valid logons.

If the logon is valid, the terminal is passed to the appropriate application program if the application is active and accepting logons. The application program is the one specified for that logon message via the interpret table, or in the case of a standard logon, it is the application program named in the message itself.

If the logon message is invalid, if the application program is not active, or if the application program is not accepting logons, the terminal operator is notified that the logon has been rejected and is invited to enter another logon message. VTAM writes error messages to the network-operator’s console if the terminal is not supported by the network solicitor, if the terminal is input only, or if the network solicitor is to be terminated.

**Network Solicitor—Modifying:** A default network solicitor is automatically included in VTAM during system generation. This network solicitor functions as described above, in “Network Solicitor—A Description”. The default network solicitor has the name NETSOL and can release terminals to requesting application programs (as explained under “Network Solicitor Release Request,” below). The default network solicitor can also be started and stopped via VTAM’s network-operator facilities.

An installation has the option of retaining this default network solicitor, of modifying the network solicitor, or of replacing the network solicitor with its own logon monitor. If the default network solicitor is to be used, the installation need only establish logon capabilities for start-stop and BSC terminals as described in “Defining Terminal-Initiated Logons,” earlier in this chapter. The network solicitor can be modified through the use of VTAM’s NETSOL macro instruction as explained below. Replacing VTAM’s network solicitor by an installation-written logon monitor is discussed below, under “Replacing the Network Solicitor”.

The network solicitor is modified by coding, assembling, and link-editing VTAM’s NETSOL macro instruction. If a NETSOL macro instruction is not assembled and link-edited to replace the default network solicitor, VTAM’s default network solicitor remains available for use.

A modified network solicitor is generated by coding and assembling VTAM’s NETSOL macro instruction. The modified network solicitor can replace, or be used in addition to, the default network solicitor. If the modified network solicitor is to be a replacement, it must be link-edited with the VTAM modules in the VTAM load module library.

Using the NETSOL macro instruction, an installation can tailor the following facets of the network solicitor:

- *Network solicitor name.* The installation can change the name of the network solicitor.
- *Messages.* The network solicitor writes messages to the terminal if unusual conditions are encountered while processing a logon. The IBM-supplied messages can be replaced by ones specified by the installation.



- *Release request.* The installation can specify whether the network solicitor should release terminals to application programs that are attempting to acquire them.
- *Password.* The installation can specify a password to be included in the network solicitor's ACB.

More detailed information on the NETSOL specifications is provided below.

**Network Solicitor's Name:** The name of the default network solicitor is NETSOL. If the modified network solicitor is to replace the default network solicitor, this name can be retained. (*Note:* Only the network solicitor that runs in VTAM's partition or private address space can use the name NETSOL.) If any other name is specified, the modified network solicitor is treated as an application program by VTAM. That is, the network solicitor must run in its own partition or private address space, the installation must supply an APPL definition statement for it, and it must be started and stopped like an application program. VTAM's start options and MODIFY command cannot be used to start or stop a network solicitor with a name other than NETSOL. See Chapter 4 for using the start options or the MODIFY commands with the network solicitor.

The name specified on the NETSOL macro instruction is the name in the automatic-logon specification for each terminal to be monitored by the network solicitor.

The load module name of the network solicitor is ISTNSCOO; this load module name must be used when modifying the default network solicitor.

**Network Solicitor Messages:** The network solicitor issues a message if any of the following conditions is encountered:

- The application program specified in the logon request is unavailable for logons. The application program is unavailable if it is inactive, closing down, or not accepting logon requests.
- The logon message is invalid; that is, it does not match any entry in the specified interpret table or (for OS/VS only) it is not in standard format.
- No interpret table is specified (DOS/VS only) or no interpret table is available for the terminal.
- The telecommunication system is closing down.
- An input error is encountered.
- The logon request is rejected by the authorization facilities of VTAM.
- The terminal is not supported by the network solicitor.

For each of these conditions, the installation can replace the IBM-supplied message with one of its own.

**Network Solicitor Release Request:** If an installation authorizes application programs to acquire terminals, the network solicitor should probably be able to release, upon request, terminals it is monitoring. If release request is specified in the NETSOL macro instruction, the network solicitor is generated with a RELREQ (release request) exit-routine like the one in the default network solicitor. Whenever this exit-routine is scheduled, the network solicitor releases the requested terminal unless a logon request is being processed. (See Chapter 5 for details on the RELREQ exit and how it is invoked and on acquiring terminals.)

**Password:** If the installation wants the ACB for the modified network solicitor to contain a password, this password must be specified in the NETSOL macro instruction. If a password is specified, the modified network solicitor is treated as an application

program by VTAM; that is, it must run its own partition or private address space, the installation must supply an APPL definition statement for it, and it must be started and stopped like an application program. VTAM's start options and MODIFY command cannot be used to start or stop a network solicitor with a password.

**Replacing the Network Solicitor:** If an installation does not want to use the IBM-supplied network solicitor, but does want a general-purpose terminal-initiated logon facility for start-stop and BSC terminals, an installation can code an application program to perform the network-solicitor functions. Such an application program would need to monitor terminals for logons and pass valid logons to the appropriate application programs. The monitoring program would be treated like an application program by VTAM: an APPL definition statement would have to be filed for it, and it would have to be activated and deactivated as an application program. (See "Activating and Deactivating Nodes," in Chapter 4, for details on activating application programs.) As an application program, though, the installation-coded monitoring program would still have access to the interpret tables via the application-program INTRPRET macro instruction.

### **Coding and Including Installation Exit-Routines**

VTAM provides two types of exits: application-program exits and installation exits. The application-program exit-routines are coded and identified in each application program that uses VTAM. These exit-routines are executed as part of the application program and are under the control of the application program. The installation exit-routines for VTAM are coded and included in VTAM as part of VTAM definition. These routines are executed as part of VTAM and are not under the control of application programs. Information on the application-program exits is provided in "Application Program Facilities," in Chapter 5. The remainder of this section provides information on coding and using the installation exit-routines, and on including them in VTAM.

VTAM provides three types of installation exits:

- An authorization exit—to validate connection, disconnection, and logon requests.
- An accounting exit—to collect accounting information.
- Logon-interpret exits—to determine the appropriate application program to receive a logon.

VTAM provides for one authorization and one accounting exit-routine. If the installation does not provide its own exit-routines, IBM-supplied modules are used. One logon-interpret routine can be coded for each LOGCHAR macro instruction, although the same routine can also be used for more than one LOGCHAR instruction. Logon-interpret routines are not provided with VTAM; if these routines are needed, they must be supplied by the installation. Each type of exit-routine is described below.

**Authorization Exit-Routine:** VTAM provides an exit that permits the installation to authorize connections between application programs and terminals. (Authorization performed in the authorization exit-routine is in addition to that performed by VTAM.) This exit is scheduled whenever a terminal is to be queued for logon to, connected to, or disconnected from an application program. (See "Application Program Concepts and Facilities," in Chapter 5, for the distinction between connection and queuing for logon.) Thus, the routine at this exit is executed whenever a connection is to be made or broken as a result of an OPNDST or CLSDST macro instruction in the application program, or whenever a terminal is to be queued as the result of a logon. (See "Characteristics of Logons," later in this chapter, for a description of VTAM logons.)

Upon entry to this exit-routine, the following information is available:

- The type of request that has been made; that is, whether the request is for a connection, a disconnection, or a logon. Also, if it is a connection request, VTAM specifies whether it is an accept or an acquire. If it is a logon request, VTAM specifies the type of logon.
- The names of the terminal and the application program to be connected, disconnected, or queued. Also, if the operation is a logon resulting from a CLSDST macro instruction with the PASS option, VTAM also provides the name of the application program issuing the pass request.

Using this input, an installation-coded authorization routine can determine whether each connection, disconnection, or logon request should be processed by VTAM. For example, each request might be compared against a predefined, installation-specified table of valid or invalid requests. The results of this examination are then returned to VTAM. If the request is determined to be valid, it is completed by VTAM. If it is invalid, it is rejected by VTAM. Output from this routine must include whether the request is valid or invalid.

The authorization routine is included in VTAM by link-editing it as a single load module into the VTAM load module library. This routine replaces an IBM-supplied module. If the installation does not replace the IBM-supplied authorization exit-routine, all connection, disconnection, and logon requests handled by this module are treated as valid.

In planning an authorization routine, a number of factors must be considered:

- The exit-routine is executed in the supervisor state under VTAM's protection key. Therefore, errors within the routine may cause damage to VTAM's control blocks and modules. Also, security violations could occur if such a routine were designed or coded by unauthorized persons, since they would have access to much of the VTAM partition or private address space.
- The exit-routine is executed under the task for which the request is being authorized (for example, the application program that issued the connection request). If the exit-routine is abnormally terminated, this task may be terminated also.
- The exit-routine is executed inline with VTAM processing. Therefore, performance may be degraded if the routine requires lengthy processing time. While this routine is being executed, no new connection, disconnection, or logon requests are processed by VTAM, and requests involving VTAM's VARY command are not processed. Therefore, system waits (such as for disk I/O) should be avoided.
- The exit-routine is notified of pass requests. (The pass option is an option of the OPNDST macro instruction; this option is used by VTAM's network solicitor and can be used by application programs.) The network solicitor uses the pass option to pass valid terminal-initiated logons to active application programs. If the installation is to use the network solicitor to monitor terminals for logons, the installation-coded authorization routine should be designed to process the validation of pass requests involving the network solicitor.
- The exit-routine is notified if VTAM's network solicitor or the Terminal Online Test Executive Program (TOLTEP) attempt to connect to a terminal. An installation-coded authorization routine should be designed to process these preemption requests. (See "Serviceability Aids," in Chapter 6, for a description of TOLTEP.)

**Accounting Exit-Routine:** VTAM provides an exit that permits an installation to maintain accounting information about connections. This exit is scheduled whenever a terminal and an application program are connected or disconnected. Thus, the routine is executed each time the application-program OPNDST or CLSDST macro instruction is issued.

Upon entry to this routine, the following information is available:

- The name of the application program.
- The name of the terminal.
- The type of request; that is, whether the operation is a connection or a disconnection.

Using this input, an installation-coded accounting routine could note and record the time a connection is initiated. Then, upon re-entry (when that connection is being broken), the routine could note the time of the disconnection. The difference between these two times reflects the approximate connection time for the terminal and that application program.

Note that connection time is an approximate value affected by such factors as the running time of the application program, the paging rate, and the operating system setting the application non-dispatchable.

The accounting routine is included in VTAM by link-editing it as a single load module into the VTAM load module library. This routine replaces an IBM-supplied module. If the installation does not replace the IBM-supplied accounting exit-routine, no accounting statistics are gathered.

In planning an accounting routine, a number of factors must be considered:

- The exit-routine is executed in the supervisor state under VTAM's protection key. Therefore, errors within the routine may cause damage to VTAM's control blocks and modules. Also, security violations could occur if such a routine were designed or coded by unauthorized persons, since they would have access to much of the VTAM partition or private address space.
- The exit-routine is executed under the task about which the accounting information is being collected (for example, the application program that issued the connection request). If the exit-routine is abnormally terminated, this task may be terminated also.
- The exit-routine is executed inline with VTAM processing. Therefore, performance may be degraded if the routine requires lengthy processing time. While this routine is being executed, no new connection, disconnection, or logon requests are processed by VTAM, and requests involving VTAM's VARY command are not processed. Therefore, system waits (such as for disk I/O) should be avoided.
- The exit-routine is notified of connection and disconnection requests involving terminals and IBM-supplied facilities. These facilities are TOLTEP, the network solicitor, the port solicitor (this facility supports switched networks), and the interface to the Telecommunications Access Method (TCAM). The installation-coded accounting routine should be designed to process requests involving these facilities.

**Logon-Interpret Routines:** Details on the logon-interpret routines are provided under "Defining Logons," earlier in this chapter. They are mentioned here with installation exits to indicate that they can be thought of as installation-level routines as opposed to application-program-level routines. The logon-interpret routines would probably be planned and coded as part of VTAM definition. Also, since these routines can be used to prohibit logon requests, they can be used in conjunction with the authorization exit-routine to control connections.

## VTAM Node Structure

This section explains the concept of nodes in VTAM. The section also discusses VTAM's hierarchy of nodes and use of nodes.

**Node—A  
Definition**

In VTAM, a node is an addressable point in the telecommunication system. The installation uses VTAM definition statements to identify all nodes. These statements not only identify an element as a node but also place that node within a hierarchical structure of nodes. All nodes are addressed symbolically; the symbolic names are also assigned by the installation at VTAM definition.

All node structures have the same general form:

```
Major Node
  Minor Node # 1
  Minor Node # 2
  .
  .
  .
  Minor Node # n
```

Thus, each node structure is a major node which contains one or more minor nodes.

*Major nodes* include:

- Sets of application programs.
- Sets of locally attached 3270s.
- NCPs for locally attached communications controllers.
- NCPs for remotely attached communications controllers.

The name of a major node is the name of a member in the VTAM definition library. This member contains the statements that define that major node and the statements can be any of the following:

- One or more definition statements defining one or more application programs.
- One or more definition statements defining one or more locally attached 3270s.
- The definition statements defining one NCP for a locally attached communications controller.
- The definition statements defining one NCP for a remotely attached communications controller.

One or more minor nodes can be defined for each major node. Each *minor node* is organized into one or more levels. The structure of a minor node depends upon its major node; these structures are explained below. The name of a minor node is the name assigned to the VTAM definition statement defining it.

See "VTAM Definition," earlier in this chapter, for more information on defining nodes.

**Node Structure—  
Application Programs**

One or more major nodes can be defined for application programs. Each major node can contain one or more minor nodes. Each minor node is represented by an APPL definition statement.

The application-program nodes that can be defined and addressed are therefore major nodes and individual programs represented to VTAM as minor nodes. Each major node can contain more than one minor node, but each minor node represents only one application program.

**Node Structure—  
Local 3270s**

The node structure for locally attached 3270s is similar to that of application programs. More than one major node can be defined for local 3270s. Each major node can contain one or more minor nodes.

Each local 3270 minor node is represented by a LOCAL definition statement. Each major node for locally attached 3270s is represented by one LBUILD definition statement and one or more LOCAL statements filed as a member of the VTAM definition library.

As with application programs, a minor node for a local 3270 represents only one 3270.

### *Node Structure— Local NCP*

One or more major nodes can be defined for each locally attached communications controller used by VTAM. A major node for a local communications controller is defined by the definition statements filed for an NCP. Thus, in the case of a locally attached communications controller, a major node is the NCP as opposed to the controller.

A major node for an NCP for a locally attached communications controller is organized as follows:

Major Node--(the NCP)  
Group  
Line  
Port                   Minor Node  
Cluster                Structure  
Terminal  
Component

A major node can have more than one minor node structure.

The port (node), the cluster (node), and the component (node) are included only for those configurations using switched lines, cluster control units, or components.

Because the minor node structure depicts a physical configuration of lines and devices, an NCP major node can contain more than one line, cluster, terminal and/or component node definition.

Minor nodes within NCP major nodes are named and specified in VTAM definition statements; thus:

- A group (node) is defined by a GROUP definition statement.
- A line (node) is defined by a LINE definition statement.
- A port (node) is defined by a TERMINAL statement.
- A cluster (node) is defined by a CLUSTER definition statement.
- A terminal (node) is defined by a TERMINAL, VTERM, or LU definition statement.
- A component (node) is defined by a COMP definition statement.

See "Network Control Program Requirements," in Chapter 7, for a description of the VTAM definition statements used to define the NCP minor nodes.

**Note:** VTAM distinguishes between remote communications controllers and remote stations in its node structure. The INNODE statement in an NCP definition deck for a locally attached communications controller represents a remotely attached communications controller. The NCP for the remotely attached communications controller must be defined as a major node in the same VTAM definition library as that containing the major-node definition for the NCP for the locally attached communications controller. There is no major-node definition in this library for the remote station. A TERMINAL statement represents a remote station,

and a remote station is a terminal (minor) node. (See Appendix C for the distinction between the remotely attached communications controller and the communications controller as part of a remote station.)

### ***Node Structure— Remote NCP***

The node structure for an NCP for a remotely attached communications controller is the same as that for an NCP for a locally attached communications controller. The only exception is that a remotely attached communications controller cannot attach another remotely attached communications controller. See “Node Structure—Local NCP,” above, for a description of the NCP node structure.

### ***Using the VTAM Node Structure***

VTAM works with nodes. Nodes are addressed, used, and manipulated by application programs via VTAM’s application-program macro instructions and by the network operator via VTAM’s network-operator commands.

Before a minor node (such as a single terminal or an application program) can be addressed by VTAM, its major node must be activated. A major node can be activated either during start processing or, thereafter, via VTAM’s VARY command.

In most cases, the activation and deactivation of major nodes are independent of each other; although: (1) A local NCP major node must be activated before its remote NCP major nodes are activated, and (2) Major nodes containing a definition of the same minor node or containing minor nodes with the same name cannot be active at the same time.

The hierarchical structure of each major node is significant to VTAM. An application program major node and a local 3270 major node each has a two-level structure (the major node with a single level of minor nodes). The NCPs, on the other hand, can have a structure of several levels (minor nodes are structured as groups, lines, etc.).

The hierarchical node structure is made of higher-level nodes and subordinate nodes. For example, an application-program minor node is subordinate to the higher-level, application-program major node. Likewise, a component node is subordinate to the higher-level, terminal node; the terminal node, in turn, is subordinate to nodes above it (such as the line or the group nodes). All minor nodes are subordinate to their higher-level, major node.

This hierarchical, or layered, structure is important when attempting to activate nodes. The importance is this: A node cannot be effectively activated until all node layers above it have been activated. See Chapter 4 for details on activating nodes.

The layered node structure enables an installation to dynamically and quickly parcel the telecommunication network to meet fluctuating requirements. Thus, the network operator can activate large segments of the network, then activate and deactivate subsections within each segment as needs change.

A major node for locally attached 3270s contains one or more local-3270 minor nodes (that is, one or more locally-attached 3270 definitions); likewise, a major node for application programs contains one or more application-program minor nodes (that is, one or more individual application-program definitions).

This collecting of minor nodes under a major node makes it easier to control the system. With major nodes an installation can activate and deactivate a set of minor nodes at one time. For example, if several application programs normally executed at the same time, they could all be defined (via an APPL statement for each ACB to be opened) in the same major node. To start these application programs, only one activation request would be

required—the one for the major node. In contrast, if each application program is defined as a minor node under a different major node, an activation request would be required for each major node.

The same minor node can be defined in more than one major node. (There is a restriction, however: Only one major node containing a definition of the same minor node can be active at one time.) Also, minor nodes of the same type (for example, local 3270s) need not all be defined in the same major node. (That is, it is possible to create several major nodes for local 3270s, each major node containing a subset of all the local 3270s in the system.)

The ability to create different combinations of minor nodes can be used not only to control the telecommunication system but also to assist in testing and expanding it.

For example, to help in controlling the network, a local 3270 could be defined in a major node that is activated when activity on the system is low. This major node would define only a subset of all the locally attached 3270s. The terminal could also be defined in another major node (defining all of the local 3270s) that is activated when telecommunication activity is high.

In addition, to help with testing, this same terminal could also be in a third major node that defines some terminals that are to be added to expand the system. This major node would be activated when testing the expanded system, or (after testing is completed) to support that expansion.

For more information on naming nodes, see “VTAM Naming Considerations,” below. See Chapter 4 for information on VTAM commands and on activating and deactivating nodes.

### *Naming Nodes*

All nodes must be named. The name of each minor node is supplied on its definition statement. The name of each major node is the name of the member (in OS/VS) or book (in DOS/VS) containing the definition statements for that major node.

Node names provide the means of addressing and using a VTAM system. These names are used:

- During VTAM definition to define the telecommunication system.
- In application programs to connect with terminals.
- By the network operator to control the VTAM system.
- In VTAM messages to identify specific portions of the system.
- By, possibly, logical units and operators at start-stop and BSC terminals to log onto an application program.

A VTAM system may have a large number of nodes, particularly minor nodes, and therefore requires the generation of many node names. Considering the quantity of names and their extensive use, some care must be taken in choosing, assigning, and disseminating names.

On assigning names, the following rules apply:

- Duplicate major-node names are not permitted.
- Duplicate minor-node names are not permitted in the same major node.
- Two or more major nodes containing duplicate minor-node names cannot be active simultaneously.



- The names NETSOL, TOLTEP, and TRACE are assigned to IBM-supplied facilities and therefore cannot be assigned by the installation to nodes.

Controlling the dissemination of node names provides some control over the security of the telecommunication system. See "Telecommunication Security," in Chapter 7, for more information on using node names to protect the system.

## **VTAM Buffering**

This section explains how VTAM handles its buffers. VTAM has a number of buffer storage-pools from which buffers are dynamically obtained and released. The installation can control the number of buffers devoted to any one terminal/application-program connection and can specify the size and utilization of each buffer pool. How VTAM buffering works and the use of the buffer controls to balance a telecommunication network load are explained below.

### ***VTAM Buffers and Connected Terminals***

Data passes through VTAM buffers when it is moved between application programs and the telecommunication network. These buffers are in VTAM storage, and the sizes of the storage pools, from which the buffers are obtained, are specified at VTAM start-time.

When VTAM is executing, it automatically obtains buffers from, and returns them to, the installation-specified pools. Buffers are obtained when data is transferred into VTAM and released when data is transferred out.

In OS/VS, to keep these pools from being monopolized by any one terminal/application-program connection, VTAM enables the installation to control the number of buffers devoted to each connection. The statement that defines an application program (the APPL statement) and the statement that defines a terminal (the LOCAL, TERMINAL, COMP, VTERM, and LU statements) each contain a buffer specification. When a terminal and an application program are connected, the buffer specification for the terminal is multiplied by that for the application program. The resulting product indicates the maximum number of VTAM buffers that can be devoted to the terminal at any one time for the duration of the connection.

This buffer specification applies only to input operations. Input data is transferred to application programs only at the request of the application program; output data is transferred from the application program to VTAM when VTAM is ready to transmit the data to the terminal.

When a terminal reaches its specified buffer limit, any additional data from that terminal may be lost. Data already in the VTAM buffers should be read by the application program.

### ***Specifying Buffer Sizes and Thresholds***

In addition to the VTAM storage pools used to obtain buffers for application program data, VTAM also has pools from which it obtains storage for control blocks used to service telecommunication requests. As with the data buffers, the storage for these control blocks is obtained and released by VTAM as needed, and the installation can specify the sizes of these pools.

In OS/VS, when the installation specifies the size of the VTAM storage pools (at VTAM start-time), it can also specify a threshold for each pool. (The threshold should be less than the size of the pool; it cannot be more.) If, during VTAM execution, this threshold is reached, VTAM avoids requesting buffers from the affected pool and does not resume

requesting buffers until sufficient buffers have been returned to drop the number of allocated buffers below the threshold. While a buffer pool is operating in its threshold, VTAM activity is slowed down and requests are queued until sufficient buffers have been released to resume normal operations.

### ***Controlling VTAM Buffering***

VTAM enables the installation to control the amount of storage devoted to VTAM buffers and to control the utilization of these buffers. The installation can control the number of buffers allocated at one time for each terminal/application-program connection. By controlling this allocation, the installation can reduce the opportunity for a connection to monopolize VTAM storage pools. In addition, the buffer limit specified can be varied to meet requirements based on such factors as terminal type or teleprocessing activity (for example: inquiry-response versus batch telecommunication processing).

The installation can also control the size and the threshold value for each VTAM buffer pool. Use of the threshold specification can enable an installation to reduce the incidence of insufficient buffers resulting in lost data.

VTAM's buffer pools are resources that can be allocated. Their size can be limited, while the need for the buffers can exceed their availability. In setting up buffer pools and controlling their use, the installation should consider such factors as:

- The amount of storage to be devoted to VTAM. (The larger the pools, the more storage required.)
- The impact of VTAM slowdown on telecommunications. (When a buffer threshold is reached, requests to VTAM are often queued.)
- The relative priorities of the various telecommunication jobs. (The higher priority terminal/application-program connections might require higher buffer limits.)
- The impact of lost data. (If buffer space is not available, data can be lost.)

## **Characteristics of Logons**

This section summarizes the types of logons available in a VTAM system. It also suggests ways of enabling the system to process logoffs. "VTAM Definition," earlier in this chapter describes how logons are specified to VTAM.

### ***Logon Types***

A logon is a request by or on behalf of a terminal to be connected to an application program. When a logon is requested, VTAM queues the request to the application program. The application program must then either accept the terminal (via the OPNDST macro instruction with the ACCEPT option) to complete the connection or reject the request (via the CLSDST macro instruction) to release the terminal. If the application program has an active logon exit-routine, the exit-routine is notified of the queued logon request. If no such exit-routine exists for the program, the terminal is only queued; the application program is responsible for determining that the terminal is available and for issuing an OPNDST or CLSDST macro instruction.

VTAM provides four types of logons:

- *Automatic logon*, in which VTAM automatically logs a specified terminal (when it is activated) on a selected application program.
- *Terminal-initiated logon*, in which the logical unit or the operator at a start-stop or BSC terminal (including a locally attached 3270) enters a message causing that terminal to be logged onto a selected application program.

- *Application-program logon*, in which an application program in the host CPU can issue the VTAM SIMLOGON macro instruction to cause a terminal to be logged onto that program, or it can issue the VTAM CLSDST macro instruction with the PASS option to cause the terminal to be logged onto another application program.
- *Network-operator logon*, in which the VTAM network operator can use a VTAM command to cause a specified terminal to be logged onto a specified application program.

The different types of logons provide varying degrees of resource sharing and of installation control over the network. They also require varying degrees of application-program involvement and of installation preparation.

Each type of logon is discussed below in more detail. See “Application Program Concepts and Facilities,” in Chapter 5, for details on queuing logon requests, connecting, and disconnecting.

### Automatic Logon

To enable VTAM to process automatic logons, the installation must specify the name of the application program to be notified whenever a particular terminal is available for connection. This specification is made when the terminal is defined at VTAM definition. The application-program name is the name assigned to the APPL definition statement for an application program. This name is specified in a GROUP, LINE, CLUSTER, VTERM, TERMINAL, COMP, or LU definition statement for remotely attached devices and in the LOCAL definition statement for locally attached devices.

If automatic logon is specified in the terminal’s definition statement, VTAM attempts to log that terminal onto the specified application program whenever the terminal is active and not connected to or queued for logon to another program.

If no program is specified for automatic logon for a terminal, that terminal can only be used:

- If an application program uses VTAM’s OPNDST macro instruction with the ACQUIRE option to request a connection with that terminal.
- Or if an application program uses VTAM’s SIMLOGON or CLSDST (with the PASS option) macro instructions to simulate a logon from that terminal.
- Or if the network operator issues a VTAM logon command on behalf of that terminal.
- Or, in the case of logical units, if the terminal initiates a logon request.

**Note:** Instead of the name of an application program, NETSOL can be specified for automatic logon. NETSOL is the name of VTAM’s logon monitor facility (the network solicitor) for start-stop and BSC terminals. Specifying NETSOL enables the network solicitor to process terminal-initiated logons. See “Terminal-Initiated Logon,” below, for more information on this type of logon.

### Terminal-Initiated Logon

To allow VTAM to process logon requests from terminals, the installation must define the requests and the logon messages. (See “VTAM Definition,” earlier in this chapter, for information on defining logons.)

In addition, for start-stop and BSC terminals, the installation indicates which terminals in the network are to be processed by the network solicitor. Terminals are so indicated by the automatic logon mechanism described above. Instead of specifying the name of an installation-coded application program, NETSOL (the name of the network solicitor) is specified in the terminals’ definition statements at VTAM definition. See Figure 3-5 for a description of how the network solicitor processes terminal-initiated logons.

The discussions provided in this publication concerning terminal-initiated logons for start-stop and BSC terminals (including locally attached 3270s) include the use of VTAM's interpret table and the network solicitor. It is possible for an installation to provide its own terminal-initiated logon facility in place of the network solicitor.

For logical units, VTAM processes terminal-initiated logons whenever the terminal is not connected to an application program. If an interpret table is specified for the terminal, the logon is matched with the table, and the logon request is passed to the application program indicated in that table. If no interpret table is specified, VTAM assumes the logon is in the standard format and passes the request to the application program explicitly named in the logon request.

#### **Application Program Logon**

Application-program logon occurs when an application program issues VTAM's SIMLOGON macro instruction or VTAM's CLSDST macro instruction with the PASS option. These instructions cause a logon request to be issued to an application on behalf of the specified terminal.

A macro instruction calling for application-program logon is processed by VTAM only if the application is authorized, during VTAM definition, to issue such a macro instruction. (See "Telecommunications Security," in Chapter 7, for a discussion on the possible impact of application-program logon to security.)

#### **Network-Operator Logon**

The final type of logon is the network-operator-initiated logon. Using a VTAM command (VARY), the network operator can specify the name of a terminal that is to be logged onto a selected application program. The network operator specifies in the command the names of both the terminal and the application program.

Besides generating a logon request on behalf of a terminal, this command temporarily modifies any automatic logon designation that may exist for the terminal. Once the command is issued, the terminal specified in that command continues to be logged on to the application program whenever the terminal is available. The newly designated automatic-logon specification remains in effect until:

- The network operator issues another logon command for that terminal and specifies another application program.
- Or the major node containing that terminal is deactivated. When the major node is reactivated, the automatic logon conditions specified at VTAM definition are in effect.

#### **Comparing Types of Logons**

Logon provides a means of notifying an application program when a terminal (or terminal operator) needs to communicate with the application, without being permanently connected to the application. VTAM offers different types of logons to enable an installation to tailor the logon facilities of its telecommunication system to its specifications. Some of these differences between the types of logons offered by VTAM are discussed below.

Who is the actual originator of the logon request is one of the more obvious differences among the VTAM logons. The logon request is actually originated from the terminal only in the case of the terminal-initiated logon. The other logons provide a means of originating a logon request *on behalf* of the terminal.

Some of the logons offered by VTAM provide for the inclusion of a logon message; some do not. Application programs using VTAM can be designed to handle only logon requests that include logon messages, only logon requests that do not include logon messages, or

either type. The following logon facilities can be used to generate a logon message for the application program's logon exit:

- Terminal-initiated logon.
- Application-program logon.

Using the terminal-initiated logon facility, a logical unit or the operator at a start-stop or BSC terminal can enter a logon message that is passed to the application program. Using the application-program logon facility, the application program issuing the logon request can specify a message in that request. In either case, the application program can use VTAM's INQUIRE macro instruction to obtain the data in the logon message. The automatic logon and the network-operator logon facilities do not include provisions for generating logon messages for the logon exit.

VTAM logons also differ in respect to VTAM definition requirements. To use the automatic logon facility or the terminal-initiated logon facility, the installation must have set up the necessary controls at VTAM definition. (See "VTAM Definition," earlier in this chapter, for details on establishing these types of logon facilities). If only standard logons are to be issued from logical units, no explicit specification for logon is necessary for VTAM. To use the network-operator logon facility, no explicit specification is required during VTAM definition. Use of the application-program logon facility requires that the application program be authorized during VTAM definition.

The completion of a logon (that is, the establishment of the VTAM connection between the application program and the terminal) can be prohibited by an installation-coded authorization exit-routine. See "Telecommunications Security," in Chapter 7, for details on controlling authorization. See "VTAM Definition," earlier in this chapter, for details on the authorization exit. See "Application Program Concepts and Facilities," in Chapter 5, for details on the logon exit and the OPNDST macro instruction and for information on queuing logon requests and completing connections.

## **Logoffs**

Logoffs from logical units can occur in two ways:

1. The logical unit, perhaps as the result of its interpretation of a work station operator command, can request the VTAM application program to disconnect it in an orderly fashion. To do this, the logical unit sends the application program a message that contains a request-shutdown indicator. This indicator tells the application program to finish any current processing, possibly send a final message to the logical unit, and disconnect the logical unit, using the CLSDST macro instruction. In this way, the VTAM application does not need to check data in each message received from the logical unit for a logoff request.
2. The logical unit can send a message, perhaps forwarded from a work station operator, that indicates a request for logoff (disconnection). This type of logoff request requires that the VTAM application program check each message received from a logical unit for a logoff request. (This is the only type of logoff request that would be received from a BSC or start-stop terminal.)

**Note:** Whatever the procedure followed by an application program for logoffs, when a terminal with an automatic logon specification is disconnected, VTAM usually attempts to queue the terminal to the application named in that specification. Some conditions can arise in which a terminal is not immediately queued as per the automatic logon specifications. For a description of these conditions see the discussion on disconnection in "Application Program Concepts and Facilities," in Chapter 5.

## CHAPTER 4. CONTROLLING A VTAM SYSTEM

This chapter describes how an installation controls VTAM. The chapter stresses the controls available to the network operator.

### Levels of Control

An installation can control VTAM in two ways:

- By the specifications it makes during VTAM definition.
- By using network-operator commands.

Using VTAM-definition and NCP-generation facilities, an installation defines and tailors a VTAM system. Defining and tailoring the system are activities that take place prior to the actual use of that system. The activities require extensive planning, and they usually have long-range effects; that is, the definition of the system and the generated NCPs are not usually subject to frequent changes.

In contrast, the network operator controls a VTAM telecommunication system between the time VTAM is started and the time it is halted. This control is provided through VTAM network-operator commands.

VTAM-definition facilities enable an installation to define the operating limitations of the telecommunication system, limitations of the telecommunication system, while the network-operator facilities enable the network operator to modify, within these limitations, the system's activity in response to varying requirements. For example, using VTAM definition facilities, an installation can define telecommunication configurations and specify valid connections; the network operator, working within these definitions, can activate and deactivate nodes and connections to control the use of the system.

The remainder of this section provides more information on VTAM's network-operator facilities. For additional information on defining a VTAM system, refer to the following sections:

- "VTAM Definition," in Chapter 3—for defining the telecommunication configuration to VTAM, for using installation exit-routines, and for filing start options.
- "Telecommunication Security," in Chapter 7—for controlling connections.
- "Operating System Requirements," in Chapter 7—for cataloged procedures.
- "Network Control Program Requirements," in Chapter 7—for NCP generation.

### VTAM Commands

VTAM's *network-operator commands* enable the network operator to monitor and control the telecommunication system. VTAM commands are a subset of the operating system commands; as such, they must be entered from a system console. Incorrect commands are rejected by VTAM's command facility, and a message specifying the error is written to the operator.

Using commands, the network operator can:

- Start VTAM.
- Stop VTAM.
- Monitor the status of the telecommunications system.

- Activate and deactivate nodes.
- Initiate requests for connections between terminals and application programs.
- Start and stop selected VTAM facilities.
- Change line-scheduling specifications.

### ***Starting VTAM***

Starting VTAM is the process of initializing VTAM and the telecommunication network prior to actively using the VTAM system. Starting VTAM can include the activation of all or only some of the nodes. It can also include the activation of selected VTAM facilities.

At the time VTAM is started, the installation tailors the telecommunication system either by entering VTAM start options through the network operator's console or by naming a data set that contains predefined start options. The data set is one of the ATCSTRxx members (members for OS/VS, books for DOS/VS) of the VTAM definition library. See "VTAM Definition," in Chapter 3, for information on predefining VTAM start options in ATCSTRxx data sets.

The VTAM start options that can be entered through the network operator's console are almost the same as those that can be predefined and filed as a ATCSTRxx member. The options that can be entered through the console include:

- Whether VTAM's network solicitor is to be activated.
- Whether VTAM's trace facility is to be activated, and if activated, for which nodes.
- Which major nodes are to be activated.
- Sizes of VTAM storage pools.
- The maximum number of NCP and locally attached 3270 major nodes that will be active at any one time.
- Whether other start options should be taken from an ATCSTRxx member.

Except for the last item (a pointer to a list of predefined start options), start options that can be specified by the network operator can also be predefined in an ACTSTRxx member.

Starting VTAM differs slightly between DOS/VS and OS/VS. Each are discussed below.

#### **Starting VTAM in DOS/VS**

VTAM is started in DOS/VS by first starting the VTAM partition and then invoking the VTAM procedure. No VTAM start options can be entered from the console in either of these two steps.

When the network operator starts VTAM, any start options in the ATCSTR00 book are processed first. If prompting was specified in the ATCSTR00 book, VTAM prompts the network operator for start options. Finally, if any errors were encountered in the start processing, VTAM prompts the network operator for final options. These options are the last to be processed before VTAM completes initialization.

#### **Starting VTAM in OS/VS**

The START command is used to start VTAM in OS/VS. This command can be entered alone or with any of the VTAM start options listed above.

When the network operator starts VTAM, any start options in the ATCSTR00 member are processed first. If the operator included any start options with the START command, these options are processed next. If no start options were specified in the START command and if prompting was specified in the ATCSTR00 member, VTAM prompts the

network operator for start options. Finally, if any errors were encountered in the start processing, VTAM prompts the network operator for final options. These options are the last to be processed before VTAM completes initialization.

### ***Halting VTAM***

VTAM can be closed down in two ways: with an orderly shutdown and with a quick shutdown. VTAM's HALT command is used for both types of shutdown. Orderly shutdown is designed to halt VTAM under normal, planned conditions. Quick shutdown is designed for emergency situations in which halting telecommunications takes precedence over loss of terminal connections and loss of data.

#### **Orderly Shutdown**

If the network operator specifies an orderly shutdown, VTAM does not prohibit connections between terminals and application programs, although new connections of application programs to VTAM are prohibited. VTAM notifies application programs of the pending shutdown by scheduling each program's TPEND exit-routine. (The TPEND exit-routine is an application program routine responsible for halting the application's teleprocessing activities when VTAM is terminating. See "Designing TPEND for the HALT Command," later in this chapter, for information on the TPEND exit-routine.)

Except for prohibiting the opening of ACBs, VTAM allows normal operation (for example, reading and writing data) until all application programs have disconnected themselves from VTAM by closing their access method control blocks (ACBs). Then VTAM deactivates all nodes and closes down the telecommunication system.

To use VTAM's orderly shutdown to its fullest, the installation should ensure that each application program has a TPEND exit-routine and that each TPEND exit-routine is designed to halt teleprocessing activity in an orderly manner and then disconnect from VTAM. If an application program connected to VTAM does not have a TPEND exit-routine, the application is not notified of the pending shutdown. This delays the halt either until the application closes its ACB or the network operator cancels the application.

#### **Quick Shutdown**

If the network operator specifies a quick shutdown, VTAM prohibits any further communication between terminals and application programs. New connections of application programs to VTAM are also prevented. Write requests that are already being transmitted are allowed to complete, but pending input or output requests are canceled. No additional input or output requests are accepted, although data already read into VTAM buffers can be read by application programs. VTAM also notifies application programs of the pending shutdown by scheduling each program's TPEND exit-routine. Having scheduled each exit-routine, VTAM waits until all application programs have closed their ACBs before it closes down the telecommunication system.

So that the quick shutdown functions properly, the installation should ensure that each application program using VTAM has a TPEND exit-routine. Since a quick shutdown is probably indicative of an emergency situation, the exit-routine should do no more than initiate a shutdown procedure.

#### **Designing TPEND for the HALT Command**

Each application program's TPEND exit-routine is scheduled whenever VTAM is about to terminate. If VTAM is terminating as the result of a HALT command, input to the exit-routine indicates whether the shutdown is orderly or quick.



For an orderly closedown, an application program can be designed to complete current processing, to notify connected terminals that the telecommunication system is closing down, and to disconnect itself from VTAM. For a quick closedown, the application should be designed to do little more than close its ACB.

If all application programs do not close their ACBs within 45 seconds, VTAM notifies the network operator. The notification is a message indicating the names of the open ACBs. The network operator can then either permit the application programs to continue processing and wait until the ACBs are closed or can use the facilities of the operating system to cancel the job containing the application programs.

Because a CLOSE macro instruction cannot be issued in an exit-routine, this routine should set an indicator that alerts the main portion of the application program to the pending closedown. In the case of a quick closedown, for example, the closedown procedure could post an event control block (ECB) and return to VTAM. This ECB should have been previously created by the main portion of the application program, and the main portion of the program should have been waiting for it to be posted. When the ECB is posted, the application program regains control and could then issue a CLOSE macro instruction for its VTAM ACB. The CLOSE request automatically disconnects any terminals connected to the application program. (See Chapter 5 for information on the TPEND exit-routine.)

### ***Monitoring VTAM Status***

The network operator monitors the status of a VTAM system by requesting and studying status information for nodes in the system. VTAM's DISPLAY command enables the network operator to request status information and to verify changes resulting from previous operator requests. This command enables the operator to request information about the following types of nodes:

- Application programs (minor nodes).
- Terminals.
- Telecommunication lines.
- Ports. (Status displays for ports are the same as for telecommunication lines.)
- Cluster control units.
- Network control programs (NCPs).

To display the status of a node, the network operator specifies the symbolic name of the node in the DISPLAY command. A minor node specified in a VTAM DISPLAY command must be part of an active major node. An NCP major node specified in a DISPLAY command must itself be active.

The following list indicates the information displayed by VTAM for each type of node:

- For an application program—Whether the application program is currently connected to VTAM, the names of terminals connected to the application program, and the names of terminals queued for logon to this application program.
- For terminals—Whether the terminal is active or inactive, the name of the application program (if any) to which the terminal is allocated (that is, connected or queued for logon), the name of the application program (if any) for which an automatic logon is specified for the terminal, a list of traces in effect for the terminal, the current specification of the device transmission limit (for polled start-stop and BSC terminals only), the device type, a count of the input/output activity and temporary errors for the terminal, and the names of the group and the line to which the terminal is assigned (for remotely attached terminals) or the channel and unit address (for a locally attached 3270).

- For telecommunication lines—Whether the line is active or inactive, the name of the group to which the line is assigned, the names of all nodes assigned to the line and an indication of those that are active, the current specifications for polling delay, negative polling limit, and NCP session limit (for polled start-stop and BSC lines only), and whether the line is switched or nonswitched, whether a line trace is active for the line. In addition, if the line is SDLC—The names of the SDLC cluster controllers that are assigned to the line and an indication of those that are active, or the names of the remote communications controllers that are assigned to the line and an indication of those that are active.
- For a cluster control unit—Whether the control unit is active or inactive, whether a trace is active for the control unit, the names of the VTAM terminals assigned to the control unit and an indication of those that are active, and the names of the line and of the group to which the control unit is assigned.
- For NCPs—Whether the NCP is active or inactive, the channel and unit address and the type of the communications controller containing the NCP (if the NCP is for a locally attached communications controller), a list of traces in effect for the NCP the load-module name of the NCP, a count of the input/output activity and of temporary errors for the communications controller in which the NCP currently resides (if the NCP is for a locally attached communications controller), and an indication of whether the NCP is for a locally or remotely attached communications controller.

*Note:* Because an open ACB is an application program to VTAM, a program can be executing in the system but not be recognized by VTAM as an application, if it does not have an open ACB for VTAM. Displays can still be requested of application-program minor nodes for which there is currently no open ACB; such a display would indicate that the application program is inactive (not connected to VTAM).

### ***Activating and Deactivating Nodes***

Before a node can be used in a VTAM system, the node must be active. The installation can control the activation and deactivation of many of the nodes in VTAM, including both major and minor nodes. All major nodes can be explicitly activated and deactivated by the network operator, and these include:

- NCPs for locally attached communications controllers.
- NCPs for remotely attached communications controllers.
- Sets of locally attached 3270s.
- Sets of application programs.

At VTAM start time, major nodes can be activated by using VTAM start options; all active major nodes are deactivated when VTAM is halted. Major nodes can also be activated and deactivated with VTAM's VARY command while VTAM is being executed.

To activate or deactivate a major node after VTAM is started, the network operator enters a VARY command that contains the name of the node and indicates whether the request is for activation or deactivation. For an activation request, the node name is the name of the member of the VTAM definition library that contains the definition statements for the node. For a deactivation request, the name is the name that was given in an activation request.

See "VTAM Node Structure," in Chapter 3, for a definition of major nodes. See "VTAM Definition," in Chapter 3, for information on filing definition statements in the VTAM definition library. See "Starting VTAM" and "Stopping VTAM," earlier in this chapter, for information on activating and deactivating nodes by other than the use of the VARY command.

Once a local 3270 or an NCP major node has been activated, some minor nodes within it can be activated and deactivated by the network operator while VTAM is running. The minor nodes that can be dynamically activated or deactivated at the request of the network operator are:

- Start-stop and BSC lines.
- Ports.
- Cluster control units.
- Terminals (both local and remote).
- Terminal components.

Except for telecommunication lines, the installation can also specifically request the activation of any of these minor nodes when VTAM is started. Start-time activation is accomplished by specifying in a minor node's definition statement that the node is to be activated automatically when its major node is activated and then by using VTAM start options to activate that major node.

Start-stop and BSC lines are automatically activated by VTAM when an NCP is initially loaded. SDLC lines are automatically activated and deactivated by VTAM, depending on whether any node on that line is active or inactive.

As noted earlier in this chapter, halting VTAM (by using VTAM's HALT command) automatically deactivates all active nodes. The orderly mode of VTAM's HALT command deactivates the nodes only when all application programs have closed their ACB's. The quick mode of the command begins to deactivate all nodes after the last application program has been notified of the pending halt, although VTAM termination is not completed until all ACB's have been closed.

To activate or deactivate a minor node while VTAM is being executed, the network operator must enter a VARY command containing the name of the node and indicating whether the request is for activation or deactivation. *Note:* A major node containing the minor node definition must be active before that minor node can be activated or deactivated.

VTAM provides two modes of deactivation for the VARY command:

- Normal deactivation.
- Immediate deactivation.

When a normal deactivation is specified, the node is not actually deactivated by VTAM until the associated application program and terminal connections have been terminated by either the application program or by the terminal operator. Queued requests for connections involving the nodes in the deactivation are dequeued; associated application programs are notified that the terminal is now inactive. No new requests for connection with these nodes are accepted. An application program connected to a terminal to be deactivated is not notified of pending normal deactivations.

Normal deactivation closes down a portion of the network without adversely affecting other telecommunication activity. Use of this mode requires planning; that is, the installation should ensure that application programs disconnect deactivated terminals in conjunction with deactivation requests from the network operator.

When immediate deactivation is specified, deactivation of the nodes begin at once. All queued requests for connections to nodes included in an immediate deactivation request are dequeued; no new requests for connection with these nodes are accepted. All input or output operations for these nodes are immediately halted, with possible loss of data.

(Data that is already in VTAM buffers prior to the deactivation can still be obtained by the application program that was connected to the terminal; data in transit to VTAM from the terminal may be lost.) If the deactivation involves a terminal connected to an application program, the application program is notified of the deactivation by the scheduling of the application's LOSTERM exit-routine; the application program must disconnect the terminal for the deactivation to complete. (See "Application Program Concepts and Facilities," in Chapter 5, for details on the LOSTERM exit-routine.)

The immediate mode provides very tight control over the network; normal mode provides less stringent control, but allows for a more orderly deactivation. Note that, for deactivation to complete, both modes require that application programs connected to terminals included in the deactivation disconnect those terminals. The difference is that normal mode does not really eliminate a terminal from the network until it is disconnected, while immediate mode eliminates it almost at once, regardless of whether it is connected.

### **Starting and Stopping an Application Program**

Starting an application program that is to use VTAM requires the network operator to do two things. The operator must (1) activate the major node containing the APPL definition statement of the application program and (2) start the job containing the application program. The major node can be activated explicitly by using the VARY command or implicitly, when VTAM is started, by using start options. The application-program job is started like any other job in the system. The order of the two operations is not in itself critical; the critical requirement is that the major node be active when the application program attempts to open its ACB for VTAM.

The major node containing the application program definition must remain active as long as the application program maintains its open ACB. If an attempt is made via the VARY command to deactivate a major node and all associated ACBs have not been closed, the command is rejected.

To VTAM, an application program is one that is defined within an active major node and that has an open ACB for VTAM. Thus, for VTAM, stopping an application program requires only that the ACB be closed. But the major node itself cannot be deactivated until all ACBs pointing to minor nodes in it have been closed.

An ACB is closed when the application program issues a VTAM CLOSE macro instruction. It is also closed by the operating system when the application program is terminated. An application program major node is deactivated by the VARY command and by the HALT command.

### **Activating and Deactivating Local 3270s**

Activating a minor node for a local 3270 causes that terminal to be allocated, if available, to VTAM. If that terminal is not available (that is, if it is allocated to another, non-VTAM user), the activation request is rejected.

Activating a major node for a set of locally attached 3270s causes those terminals, defined in the associated LOCAL definition statements as initially active, to be allocated (if available) to VTAM. VTAM notifies the network operator of any that are not available. The network operator can then use the VARY command to activate the minor nodes for these terminals as they become available.

Deactivating a minor node for a locally attached 3270 returns the terminal to the operating system. Deactivating a major node for a set of locally attached 3270s returns all active terminals in that set to the operating system. (Inactive terminals in that set are not currently allocated to VTAM.)

If an automatic logon is specified for a locally attached 3270, a logon request is queued to the application program (if it is active and accepting logons) whenever the terminal is activated. If the application program has an active logon exit-routine, the routine is scheduled.

When a locally attached 3270 is activated, it is available for connection to application programs using VTAM. When the terminal is deactivated, it is unavailable for connection through VTAM, but it is available to non-VTAM users.

#### **Activating and Deactivating an NCP**

Activating an NCP for a locally attached communications controller causes the controller to be allocated to VTAM. Until the NCP is deactivated, the communications controller remains allocated to VTAM and is not available to other users of the operating system except through VTAM facilities. Allocating a locally attached communication controller to VTAM also implicitly allocates the associated remote attachments to VTAM. (Remember that only locally attached nodes are recognized by the operating system and need to be explicitly allocated to VTAM; the remotely attached nodes are "allocated" to VTAM because they are part of the network controlled by the locally attached communications controllers.)

Activating an NCP can also initiate the loading of the the NCP into the controller. The NCP is not loaded if the NCP specified in the activate command is already loaded and in its initial state, unmodified by the network operator. If the specified NCP is not currently in the communications controller, it is loaded.

If a currently loaded NCP is to be used, it must be in its initial state; that is, its status is that as specified for the NCP in the VTAM definition library. Thus, active cluster control units, terminals, and components are only those that are specified as active in the definition statements for the NCP. (If an NCP is modified by the network operator, it is not considered to be in its initial state.) VTAM does not automatically reestablish connections between application programs and terminals, though automatic logon requests are initiated for active terminals that have automatic logon specifications. Connections must be reestablished by using the OPNDST macro instruction. Also, terminals attached to switched lines have been disconnected and must be redialed.

If the activate request results in the loading of an NCP, the status of the NCP is that defined in the definition deck filed in the VTAM definition library. Thus, lines are automatically activated, and active cluster control units, terminals, and components are those defined as active in the VTAM definition statements for the NCP. Automatic logon requests are initiated for active terminals with automatic logon specifications.

Deactivating an NCP for a local communications controller returns the communications controller to the operating system. It does not delete the loaded NCP. The installation is responsible for loading another control program if the current one is not acceptable for the next user.

Consideration must be given to activating and deactivating a remotely attached communications controller. Terminals attached to a remote communications controller are not available for use by VTAM until the remote controller has been activated. To activate the remote communications controller, both NCPs must be activated: first the NCP for the locally attached communications controller and then the NCP for the remotely attached communications controller. Before deactivating a local communications controller in a VTAM system, the network operator does not have to deactivate remotely attached controllers. Deactivating an NCP for a local communications controller automatically deactivates any remotely attached communications controllers. *Note:* Activating an NCP for a remotely attached communications controller automatically

activates the line connecting the remote controller to the local controller; deactivating that NCP deactivates the line.

If the NCP contains PEP, activation and deactivation requests may impact emulation processing. Activating an NCP with PEP causes the entire NCP to be loaded (including both the emulation functions and the network control functions); although, deactivating the NCP through the VARY command makes the NCP and the communications controller inactive only for VTAM. This deactivation does not halt emulation processing in the controller.

VTAM's activation and deactivation requests for telecommunication lines may also have some impact on emulation mode. VTAM controls the assignment of lines that can be reassigned between network control and emulation modes. Lines that can be reassigned are automatically assigned to emulation mode unless they are activated by VTAM. Upon being activated, lines that can be reassigned but are currently assigned to emulation mode are reassigned to network control mode (except those lines being used by emulation). When they are deactivated, the lines are returned to emulation mode. See "Network Control Program Requirements," in Chapter 7, for more information on controlling an NCP with PEP.

#### **Activating and Deactivating Remote Attachments**

Activating and deactivating a minor node for a remote attachment (such as a terminal, cluster control unit, or line) does not affect the allocation of that attachment to VTAM. Remote attachments remain implicitly allocated to VTAM as long as the NCP for the locally attached communications controller is not deactivated by VTAM.

For VTAM to treat a terminal as active (that is, for VTAM to permit a terminal to be connected to or queued to an application program), the associated line and cluster control unit (if any) and the terminal must all be active. When the NCP is initially loaded, cluster control units, terminals, and components are activated as specified in the definition statements for those nodes. Start-stop and BSC lines are automatically activated when an NCP is initially loaded. SDLC lines are automatically activated and deactivated by VTAM, depending upon whether any remote node on that line is active or inactive. (See "Network Control Program Requirements," in Chapter 7, for information on defining lines, cluster control units, and terminals.)

Because active application programs can be connected to only active terminals, it is the activation of these two nodes that is of the most concern. Activating and deactivating application programs and locally attached terminals are discussed above. The following describes the activation and deactivation of remotely attached terminals.

Although an application program can be connected to an active terminal, the connection can only be completed if the terminal is accessible through an active cluster control unit (if any), an active line, and an active NCP. (If the terminal is accessible, as defined in its `TERMINAL` or `VTERM` definition statement, through more than one line, at least one of the lines must be active for connections to be completed.) Thus, a remote terminal is treated as active (that is, connected to or available for connection to an application program) only if all nodes in a valid path to the terminal have been activated.

Deactivating the NCP, the line (or lines in the case of switched networks), the cluster control unit (if any), or the terminal effectively deactivates the terminal by making it unavailable for connection to any program using VTAM. Each of these nodes is deactivated by using VTAM's `VARY` command with the deactivate option. To reactivate a terminal, the `VARY` command with the activate option must be issued for the node that was deactivated.

### *Example of Deactivating and Reactivating a Remotely Attached Terminal*

Assume an active BSC terminal in a switched network is connected to an active cluster control unit which is in turn accessible via either of two active lines. The terminal can be deactivated by using the VARY command to deactivate any of the following:

- The NCP.
- *Both* lines.
- The cluster control unit.
- The terminal.

Reactivating the terminal depends on the node that is specified in the deactive request. To reactivate the terminal, a VARY activate request specifying the same node must be entered:

- If the NCP was specified in the deactivation request, activate the NCP.
- If both lines were specified in the deactivation request, activate at least one of them.
- If the cluster control unit was specified in the deactivation request, activate the control unit.
- If the terminal was specified in the deactivation request, activate the terminal.

If a terminal is effectively deactivated by deactivating the NCP, the terminal can only be reactivated automatically when the NCP is reactivated if the following conditions are met: The terminal and cluster control unit (if any) are defined via VTAM definition statements as being initially active when the NCP is loaded.

Note that a deactivate request addressed to a terminal affects only that terminal, while a deactivate request addressed to an NCP or a cluster control unit affects all attached, active terminals. A deactivation request addressed to a line prohibits access to active terminals only through that line. If those terminals have access through other active lines, they have not been deactivated, although access to them has been restricted. If a terminal has access through only one line, deactivating that line deactivates the terminal.

Special considerations apply when activating and deactivating logical units. See "Activating and Deactivating Teleprocessing Subsystems," later in this chapter.

Activation and deactivation are essentially the same for all remotely attached terminals, whether they are attached to a local communications controller or to a remote communications controller. The only difference is that a terminal attached to a remote communications controller depends on the status of a greater number of nodes to complete an active path to an application program. In addition to the status of the NCP of the local communications controller, a terminal attached to a remote controller depends on the status of the following nodes associated with that remote controller:

- The remote communications controller itself.
- The line or lines and cluster control unit (if any) connecting the terminal to the remote controller.

Although there are more nodes in a path between an application program and a terminal attached to a remote communications controller, the same factor controls the active status of that terminal: namely, all nodes in the path must be active.

## **Activating and Deactivating Teleprocessing Subsystems**

Teleprocessing subsystems supported by VTAM must be loaded (IPLed) before they can be activated by VTAM. Thus, a 3601 controller, for example, must be IPLed before it can be activated, and the controller must be active before any associated logical unit can be activated.

Activation and deactivation of SDLC lines are performed automatically by VTAM. Because teleprocessing subsystems are attached via SDLC lines, the network operator is not responsible for activating and deactivating these lines. SDLC lines are activated automatically when a remote node (such as a SDLC cluster controller or a logical unit) on that line is active, and they are deactivated automatically when all remote nodes on the line are deactivated.

## **Special Consideration for Activation**

The activation of nodes has special implications that, in part, result from the way VTAM controls its system. When VTAM activates a major node, it builds a segment of a table containing entries which describe the minor nodes defined for that major node. (When the major node is deactivated, the table segment for that node is deleted from main storage.) VTAM uses the entries in the table to represent the actual minor nodes. It is these entries that VTAM allocates to users; VTAM does not really allocate the node the entry represents but, instead, retains the ownership of all the nodes. In most instances, the status of the minor node and of its representative table entry is the same; thus, usually, no distinction need be made between a node and its table entry. The activation of application programs and the activation of start-stop and BSC terminals (including locally attached 3270s) are two instances in which this distinction can manifest itself.

The implications of an active application program and of an active terminal are discussed below. No distinction is made in this discussion between activation that occurs automatically as a result of VTAM definition options or dynamically because of network-operator requests.

**An Active Application Program:** In respect to application programs, VTAM activates only major nodes. Once an application program major node is active, each table entry for a minor node within it is available to form a connection between VTAM and an actual application program. This connection is made when an ACB (pointing to one of these entries) is successfully opened. A table entry for an application program minor node can be used to form a connection with VTAM by only one application program at a time; that is, it can be pointed to by only one open ACB at a time.

VTAM's VARY command and the start options for activating nodes activate only the major nodes for application programs. The actual starting of the application program and the opening of the ACB are responsibilities of the installation. When the ACB is open, VTAM treats it and its associated table entry as an active application program.

**An Active Terminal:** When VTAM activates a terminal, it performs two activities: VTAM (1) indicates in the terminal's table entry that the terminal is active and (2) either transmits an activation command to the NCP (for remotely attached logical units) or obtains the terminal from the operating system (for locally attached terminals). When an application program connects with a terminal, VTAM connects the application program to this table entry, while retaining the ownership of the terminal itself.

In the case of start-stop and BSC terminals, the physical status of the terminal can differ from that of its table entry; for example, a table entry can be marked active even though the terminal it represents has not been turned on or is not even physically in the network. But, because VTAM allocates the table entry to the application program when completing a connection request, an application can connect to a terminal that is not physically part of the network. Connection is possible in this case if the table entry is marked active and a defined path has been activated.



If connection is requested to a nonexistent (but defined) terminal, the connection is completed, but no data transfer can be completed. These conditions enable an installation to include, in the definition of the network, resources that are not physically available but will be added at a later date. By doing this, an installation can avoid redefining the network or recoding application programs for the addition of minor nodes to the telecommunication system.

For logical units, the status of the table entry and the terminal it represents must agree. Thus, when a logical unit is activated, it must be physically active before its table entry can be activated and made available for use by an application program. When VTAM activates a start-stop or BSC terminal (including locally attached 3270s), an activate request for a start-stop or BSC terminal completes even if no other node in the terminal's path is active except the NCP. (Remember that a terminal without an active path is still not available for connection to an application program; in other words, the terminal is effectively inactive for application programs.)

### ***Initiating Requests for Connections***

In addition to its use in activating and deactivating nodes, VTAM's VARY command can be used to initiate connection requests on behalf of terminals. This logon option is used to initiate VTAM's network-operator logon.

To initiate a network-operator logon, the network operator enters a VARY command containing the names of the terminal and the application program to be connected. VTAM then processes the command as though it were an automatic logon for the terminal; that is, VTAM notifies the application program that a logon request was received from a terminal, but the application program must then accept the request before connection is completed. (See "Application Program Concepts and Facilities," in Chapter 5, for details on connecting application programs and terminals.)

In addition to initiating a logon request, this option of the VARY command alters the automatic-logon specification for the terminal. For example: If the network operator initiates a logon request on behalf of terminal 1 for application program A, terminal 1 is initially logged onto that program. Thereafter, whenever terminal 1 is available, it is automatically logged back onto application program A. This automatic-logon specification for terminal 1 remains in effect until the network operator either deactivates the major node containing the terminal definition or enters a new logon request on behalf of the terminal.

Because the network-operator logon modifies the automatic-logon specification, care should be exercised when using the logon facility of the VARY command. See "Characteristics of Logons," in Chapter 3, for more details on automatic and network-operator logons.

### ***Starting and Stopping VTAM Facilities***

Some of the facilities in VTAM need not be active continuously. VTAM allows the network operator to start and stop these facilities selectively.

Using VTAM's MODIFY command, the network operator can start the following VTAM facilities:

- Network solicitor.
- Trace facility.
- Print-trace utility program.
- Dump utility program.
- Teleprocessing Online Test Executive Program (TOLTEP).

The MODIFY command can be used to stop these facilities:

- Network solicitor.
- Trace facilities.

The network solicitor and the trace facilities can also be activated via VTAM start options. See “Starting VTAM,” earlier in this chapter, for more information on activating these two facilities at start time.

### **Starting and Stopping the Network Solicitor**

The MODIFY command can be used to start the network solicitor only if the network solicitor (1) is the default network solicitor or (2) was modified via the NETSOL macro instruction (with the name NETSOL). See “VTAM Definition,” in Chapter 3, for descriptions of the network solicitor and of the NETSOL macro instruction.

Activating the network solicitor causes all appropriate available terminals to be automatically logged onto it. Appropriate available terminals are terminals that meet all the following requirements:

- They are active but not connected, nor queued for connection, to another program.
- The automatic-logon specification in their node definition indicates the network solicitor.

As long as the network solicitor remains active, it continues to accept appropriate available terminals. It attempts to solicit logon requests from these terminals and passes valid logon requests to the appropriate application programs.

Deactivating the network solicitor with the MODIFY command causes the network solicitor to complete handling all terminal-initiated logon requests in process and to disconnect all other terminals connected to it. No additional automatic logons are accepted by the network solicitor until it is reactivated.

### **Starting and Stopping Traces**

The MODIFY command can be used to start or stop traces for NCP major nodes as well as for the following minor nodes:

- Telecommunication lines.
- Cluster control units.
- Terminals (locally and remotely attached).
- Components.

If the node specified in the command is a telecommunication line, the network operator can only request an NCP line trace. For all other nodes, either the buffer trace or the I/O trace can be requested.

Nodes included in a start-trace request are traced only when active. Deactivating a node stops any traces for it. If the node is reactivated and the trace is still needed, the trace request must be reissued. (See “Serviceability Aids,” in Chapter 6, for detailed descriptions of VTAM traces.)

### **Starting the Trace-Print Utility Program**

In DOS/VS, VTAM provides a utility program to selectively edit and print trace records accumulated by VTAM’s trace facility. This utility program is also started by an option of the MODIFY command. Once this utility has been started by the MODIFY command, it prints all trace records in the trace data set that the operator specified to be printed. Tracing is suspended for the affected nodes until the printing has been completed.

In OS/VS1, VTAM traces are printed by using the operating system's HMDPRDMP service aid. Refer to the publication *OS/VS1 Service Aids*, GC28-0665, for information on HMDPRDMP. In OS/VS2, VTAM traces are printed by using the operating system's AMDPRDMP service aid. Refer to the publication *OS/VS2 System Programming Library: Service Aids*, GC28-0633, for information on AMDPRDMP.

### **Starting VTAM's Dump Utility Program**

Another option of VTAM's MODIFY command enables the network operator to initiate the execution of the NCP dump utility program tailored for a VTAM environment. Using this option, the network operator specifies the NCP to be dumped. VTAM initiates loading of the NCP dump utility program into the communications controller. The dump utility program then provides the requested dump. An operating system print-utility can be used to print the dump.

Caution should be exercised when requesting a dump of an NCP. The dump program erases part of the NCP, with or without PEP, requiring that the control program be reloaded if it is to be used after the dump is taken. Thus, in the case of an NCP with PEP, the emulation function, as well as the network control function, is affected by the dump request.

If an NCP dump is to be initiated by VTAM, the dump data set, LUB name in DOS/VS, must have been specified in the NCP's PCCU statement during VTAM definition. See "Network Control Program Requirements," in Chapter 7, for a description of the PCCU statement.

### **Starting TOLTEP Testing**

The Teleprocessing Online Test Executive Program (TOLTEP) is a component of VTAM designed to provide the testing of lines and devices in the network. TOLTEP testing can be initiated from the network-operator's console in either of two ways: by using VTAM's MODIFY command or by using the logon option of the VARY command.

TOLTEP testing can be initiated from other than the operator's console; in this case, testing is requested by an operator at an active terminal in the VTAM network. The network operator is notified of the request and must authorize the request before the testing can continue.

See "Serviceability Aids," in Chapter 6, for a detailed description of TOLTEP and how it is controlled by the network operator.

### **Changing Line-Scheduling Specifications**

VTAM permits the network operator to change the following line-scheduling specifications for polled, nonswitched start-stop or BSC lines:

- Polling delay.
- Negative poll response limit.
- NCP session limit.
- Device transmission limit.

The first three specifications are made on the NCP's LINE statement. The last specification is made on the NCP's TERMINAL statement. Refer to the *NCP Generation* publication for a description of these parameters and their functions.

The line-scheduling specifications are changed by the network operator by using an option of VTAM's MODIFY command.

A line-scheduling change remains in effect until the communications controller is reloaded. If the reloading employs NCP restart (as the result of an NCP error), the change

remains in effect for the reloaded NCP. If the reloading does not employ restart, the line-scheduling parameters specified for NCP generation are reinstated.

## Considerations for Network-Operator Control

To use VTAM's network-operator facilities effectively, the installation must establish control procedures and provide information to the network operator. In general, the network operator must be familiar with the configuration of the telecommunication network, know how to manipulate VTAM to attain the installation's teleprocessing objectives, and know what actions to take when problems are encountered in the system. Planning by the installation should include the following areas:

- *Names of nodes and data sets.* Because symbolic node names are the elements manipulated by VTAM, the network operator must know the names of all nodes (major and minor) that are to be used in VTAM commands or that can be received in VTAM messages. The operator should also know the names and uses of all VTAM data sets. (See "VTAM Node Structure," in Chapter 3, for a discussion on naming nodes. See "VTAM Data Sets," in Chapter 7, for a description of the data sets used by VTAM.)
- *Relationship between the names of VTAM application programs and job names.* Application programs are identified to VTAM through APPL statements and ACBs. For VTAM, the name of an application program is the name of an APPL statement; for the operating system, the job name or the step name is the program name. The network operator needs to know how each application-program name in VTAM relates to job names or step names.
- *Relationship between the symbolic and the physical network.* Because VTAM commands function differently for major nodes than they do for minor nodes, the network operator should know which nodes are major and which are minor. Also needed is a knowledge of the hierarchical structure of the NCP nodes and of the relationship between the symbolic names and the physical units they represent.
- *Impact of VTAM on emulation in an NCP with PEP.* Although VTAM does not use emulation mode, the access method can impact the operation of emulation mode in an NCP with PEP. (See "Network Control Program Requirements," in Chapter 7, for a description of VTAM's impact on emulation mode.)
- *Switched-network support.* Controlling a switched network requires an understanding of how such a network is defined. See "Network Control Program Requirements," in Chapter 7, for details on defining and controlling switched lines and terminals that use them.
- *Storage sizes.* When the operating system is IPLed, the VTAM region or partition size can be selected. VTAM also allows the network operator to make storage-pool specifications. If these two specifications are not predefined to the system or to VTAM, the network operator needs the detailed storage information.
- *Procedures.* An installation may also want to establish procedures for the operator to follow. The procedures can be for both planned normal operations and contingencies. The procedures might cover: when and how to start and stop VTAM, application programs, and the network solicitor; when, how, and which terminals and other nodes to activate and deactivate; and what actions to take when network errors are encountered, including what traces to activate, how to activate when and how to dump an NCP.

The above list is not meant to be exhaustive because each installation has different requirements. Instead, the list is intended to call attention to the need for setting up procedures and familiarizing the operator with the network.



## CHAPTER 5. VTAM APPLICATION PROGRAMS

The purpose of this chapter is to introduce VTAM application program concepts and facilities so that an installation can begin to plan and organize its VTAM application programs.

The first section of this chapter provides an overview of VTAM application programs and a summary of VTAM macro instructions. The second section explains VTAM programming concepts and facilities. The third section explains in more detail the VTAM language and its use. The final section shows and discusses the general logic of two sample application programs.

### A VTAM Application Program Overview

This section provides an overview; it views the VTAM application program from the perspective of an installation's telecommunication system. It summarizes VTAM macro instructions, and it relates VTAM application programs to those of TCAM and BTAM.

#### *The VTAM Application Program in Relation to the VTAM System*

Figure 5-1 shows where a VTAM application program fits into a teleprocessing system. The numbers in Figure 5-1 refer to major elements in the system. VTAM application-program connection to start-stop or BSC terminals is possible but is not shown. The following discussion is keyed to Figure 5-1.

#### **The VTAM Application Program 1**

In a teleprocessing system, the two main activities are the processing of data and the transmission of data. A VTAM application program uses VTAM macro instructions to transmit data between the host CPU and the telecommunication network, and it uses other instructions to process data. VTAM permits the separation of the processing part of an application program from the telecommunication part. (If the parts are separate, an interface between them would need to be defined.) This separation of function allows each part to be created separately and means that changes or additions to one part do not affect other parts. The processing part of a VTAM application program may be written in a higher-level language, such as PL/1; the telecommunication part, using VTAM macro instructions, is written in assembler language. This chapter, describing the concepts, facilities, and language of VTAM, concerns itself mainly with the telecommunication part of the VTAM application program.

As shown in Figure 5-1, more than one VTAM application program can be concurrently sharing the resources of a VTAM system. For example, each of the VTAM application programs in Figure 5-1 can communicate with different logical units on the same transmission line; because the actual transmission is managed for the programs, neither program is aware that line sharing takes place or of the line's identity.

#### **The Processing Part 2**

This part of the program can be written in assembler language or in a higher-level language, such as PL/1 or COBOL. If written in assembler language, it can be interleaved with the telecommunication part of the program. More commonly, as shown in Figure 5-1, it is separate and requests telecommunication services by calling or branching to the telecommunication part of the VTAM application program.

The function of the processing part of the application program is to perform the data manipulation in response to a request from a remote location. Such data manipulation might include updating a data base, obtaining information from a data base, or formatting information to be displayed for an operator at a terminal. Because both the logical unit and the application program in the host CPU are programmed, it is possible to distribute

Host CPU

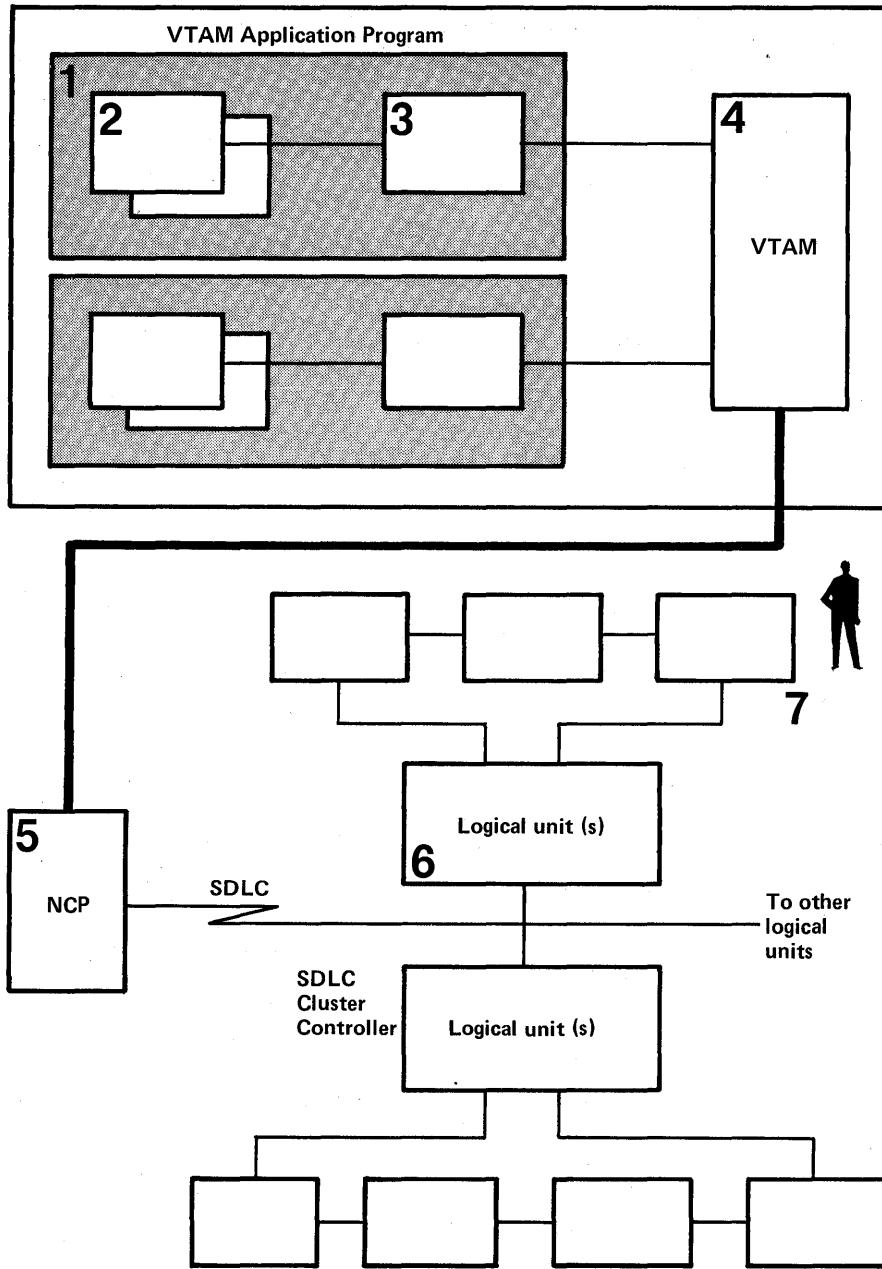


Figure 5-1. VTAM Application Programs in Relation to the Telecommunication System

the data processing function between these two nodes. For example, the application program in the host CPU could obtain data from the data base while the logical unit could format the data for display to the terminal operator.

Each installation decides which processing functions are to be performed by the processing part of the VTAM application program and which are to be performed by an application program that is part of a logical unit, such as an application program in a 3601 Finance Communications Controller.

This part of the VTAM application program contains macro instructions and associated control blocks used to connect and communicate with logical units that have been defined to VTAM. Chapter 5 is concerned primarily with this part of the VTAM application program.

## VTAM 4

VTAM controls the VTAM telecommunication system. Logical units are defined as part of the VTAM system during the VTAM definition process and are then activated by start procedures or network operator commands. The VTAM application then requests connection (on its own initiative or as the result of a VTAM-processed logon request) to one or more active logical units. Once connected, the program requests VTAM to perform data transfer with logical units. In addition to building channel programs, VTAM performs such services as input and output data buffering, automatic scheduling of application program exit-routines, and sequence numbering of outbound messages. (These facilities are discussed further in this chapter.) VTAM requests the operating system to execute channel programs it has built; the channel programs result in communication with local 3270s or with remote logical units through a locally attached communications controller.

## The Network Control Program 5

On receiving the input or output requests and associated data or information, the network control program (NCP) in the 3704 or 3705 Communications Controller does what is required to communicate with logical units on telecommunication lines. Many functions previously performed by the access method or application program are now performed by the communications controller; for example, scheduling line activity, retrying transmission-errors, and collecting error statistics. Communication with logical units is performed using SDLC.

## The Logical Unit 6

The VTAM application program communicates primarily with program logic (a logical unit) in an SDLC cluster controller rather than with static, non-programmable devices. The meaning of a logical unit generally depends on the particular telecommunication subsystem; for example, in a 3600 Finance Communication System, a logical unit is synonymous with a work station.

VTAM provides application programs in the host CPU with a set of macro instructions to communicate with logical units. This set of instructions enables applications to make use of the programmed function in logical units.

Although not shown in Figure 5-1, VTAM application programs also communicate with certain terminals on start-stop and in BSC lines. VTAM provides another set of macro instructions to communicate with these terminals.

Additionally, VTAM application programs can communicate with 3270 s (attached both remotely, via BSC lines, or locally) as though the terminals were logical units. Thus, the same set of macro instructions can be used to communicate with 3270 s and logical units.

An installation defines which processing functions take place in the teleprocessing subsystem and which in the VTAM application program in the host CPU. An installation must coordinate the writing of these programs so that they can work together.

## The Terminal Operator 7

Since a VTAM application program communicates with a logical unit rather than with a terminal operator at a physical device, the VTAM application program does not need to know about many terminal operator actions. The logical unit determines whether and how data received from a terminal operator goes to the VTAM application program and whether and how data received from the VTAM application program goes to the terminal operator.

## *A Summary of VTAM Macro Instructions*

The telecommunication part of a VTAM application program uses IBM-provided macro instructions to request VTAM services. These macro instructions, summarized below, are referred to further in this chapter in "Application Program Concepts and Facilities" and discussed in more detail in "The VTAM Language."



## Connection Macro Instructions

These macro instructions connect and disconnect an application program with VTAM and with logical units and start-stop and BSC terminals.

### OPEN/CLOSE

Connects and disconnects a VTAM application program to or from a VTAM system.

### OPNDST/CLSDST

Connects and disconnects a logical unit or BSC or start-stop terminal to and from a VTAM application program.

## Communication Macro Instructions

These macro instructions request and control the transmission of data between an application program and its logical units.

### RECEIVE

Requests that input received from a logical unit be passed to the VTAM application program.

### SEND

Requests that output be sent to a logical unit from the VTAM application program.

### RESETSR

Changes the mode of receiving input from a particular logical unit. The modes are continue-any mode (have input from the logical unit satisfy an outstanding RECEIVE that specifies input from any logical unit) and continue-specific mode (have input satisfy a RECEIVE that specifies only that particular logical unit). RESETSR also cancels outstanding RECEIVES that request input from the specific logical unit.

### SESSIONC

Initializes and controls the flow of information between the VTAM application program and a logical unit.

(Communication macro instructions for BSC and start-stop terminals are not included here; they are discussed later in this chapter.)

## Control-Block Macro Instructions

The two following categories of macros are used to build and manipulate VTAM control blocks. Figure 5-2 shows the relationship between the VTAM control blocks when a logical unit is being connected to an application program.

### *Declarative Macro Instructions*

These macro instructions are used to assemble control blocks when the application is assembled. The macros are also the names of the VTAM control blocks themselves.

### ACB

Defines the characteristics of a VTAM application program to VTAM.

### EXLST

Defines the list of exit-routine addresses that VTAM is to use for notifying the program when certain conditions occur, such as a request for logon (connection) to the program from a logical unit.

### NIB

Defines information pertaining to a particular logical unit; this information is stored by VTAM for use by VTAM throughout the time that the logical unit is connected.

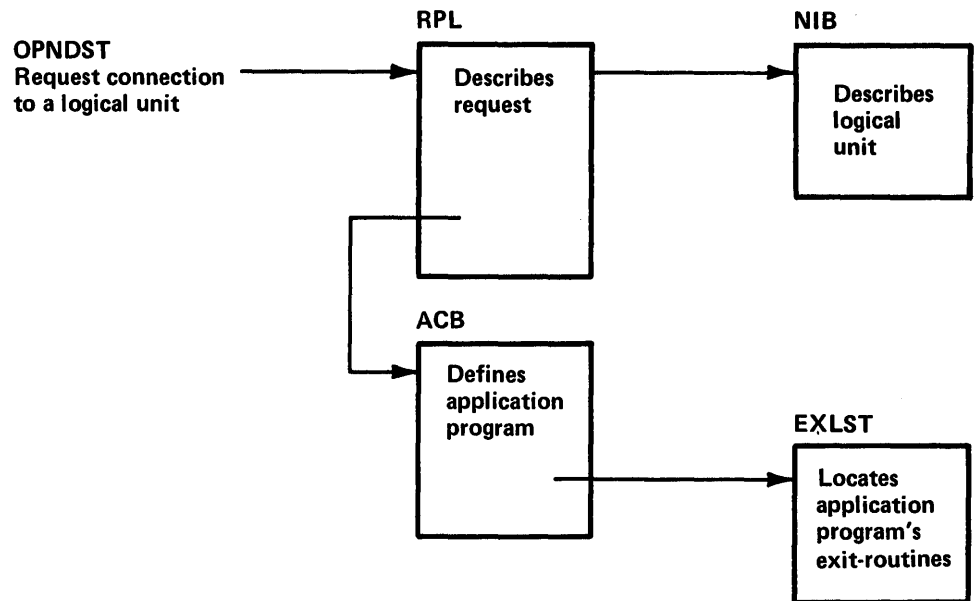


Figure 5-2. Relationship of VTAM's Control Blocks in an Application Program

#### RPL

Defines the parameters and contains the results of a particular operation (for example, OPNDST or SEND) that is requested.

#### *Manipulative Macro Instructions*

These macro instructions are used to dynamically build and manipulate control blocks during the execution of the application program.

#### GENCB

Builds and can initialize an ACB, EXLST, NIB, or RPL control block during program execution.

#### MODCB

Changes the contents of a control block field to a designated value.

#### SHOWCB

Moves the contents of a control block field to a designated area.

#### TESTCB

Tests the contents of a control block field and sets the program status work (PSW) condition code accordingly.

The VTAM user can also gain access to VTAM application program control blocks by using DSECT and other assembler language instructions.

#### **Support Macro Instructions**

These macro instructions request additional VTAM services.

#### EXECRPL

Requests VTAM to perform an operation that is currently specified in a designated RPL. EXECRPL can be used instead of using other RPL-based macro instructions such as OPNDST, SEND, and RECEIVE, or, after an unsuccessful operation, when reissuing a request.

#### CHECK

Checks and, if necessary, awaits completion of a requested operation.

#### INQUIRE

Requests information from VTAM, such as the status of another application program.

#### INTRPRET

Provides a way to obtain information from an installation-defined interpret table.

#### SETLOGON

Tells VTAM when to permit logon requests for the VTAM application program.

#### SIMLOGON

Allows the VTAM application program itself to initiate a logon request. If necessary, it also requests the current owner of the logical unit to release it.

### *A Brief Comparison with TCAM and BTAM*

The following is a brief comparison of programs that use VTAM macro instructions with those that use TCAM or BTAM macro instructions.

#### **VTAM Compared with TCAM**

TCAM provides services for handling telecommunication input/output at two different times: by the telecommunication part of a program (a TCAM message control program, including its message handler) and by the processing parts of a program (TCAM application programs). A TCAM programmer uses TCAM macro instructions to write a message control program. A message control program queues input messages until they are requested by a TCAM application program. It queues output messages received from a TCAM application program until requested by another application program or until they can be sent to a telecommunication device. Queuing can be in main storage or on a direct access storage device. The programmer uses GET/PUT and READ/WRITE macro instructions in the TCAM application program to request the services of the message control program.

VTAM provides only direct telecommunications services for the VTAM application program; the VTAM application programmer must create his own control blocks and logic for the queuing of data, if necessary, and must provide his own interfaces between the telecommunication and processing parts of the program. Figure 5-3 shows how TCAM relates to VTAM.

VTAM services are available to application programs using TCAM; this is described in "Other Telecommunication Access Methods," in Chapter 7.

#### **VTAM Compared with BTAM**

Figure 5-4 shows some of the major similarities and differences between VTAM application programs and BTAM application programs. VTAM application program characteristics shown in Figure 5-4 are discussed in more detail further in this chapter.

## **Application Program Concepts and Facilities**

This section introduces and explains VTAM programming concepts and facilities and relates them to the VTAM language (macro instructions and operands).

### *Introduction to the Concepts and Facilities*

VTAM's primary purpose is to provide communication between the application program and a terminal in the telecommunication network. (A terminal is generally a logical unit, but it can also be a BSC or start-stop terminal.) VTAM also provides a means by which the installation can establish a system of dynamically allocating logical units to the application programs. The concepts and facilities described in this chapter all deal with

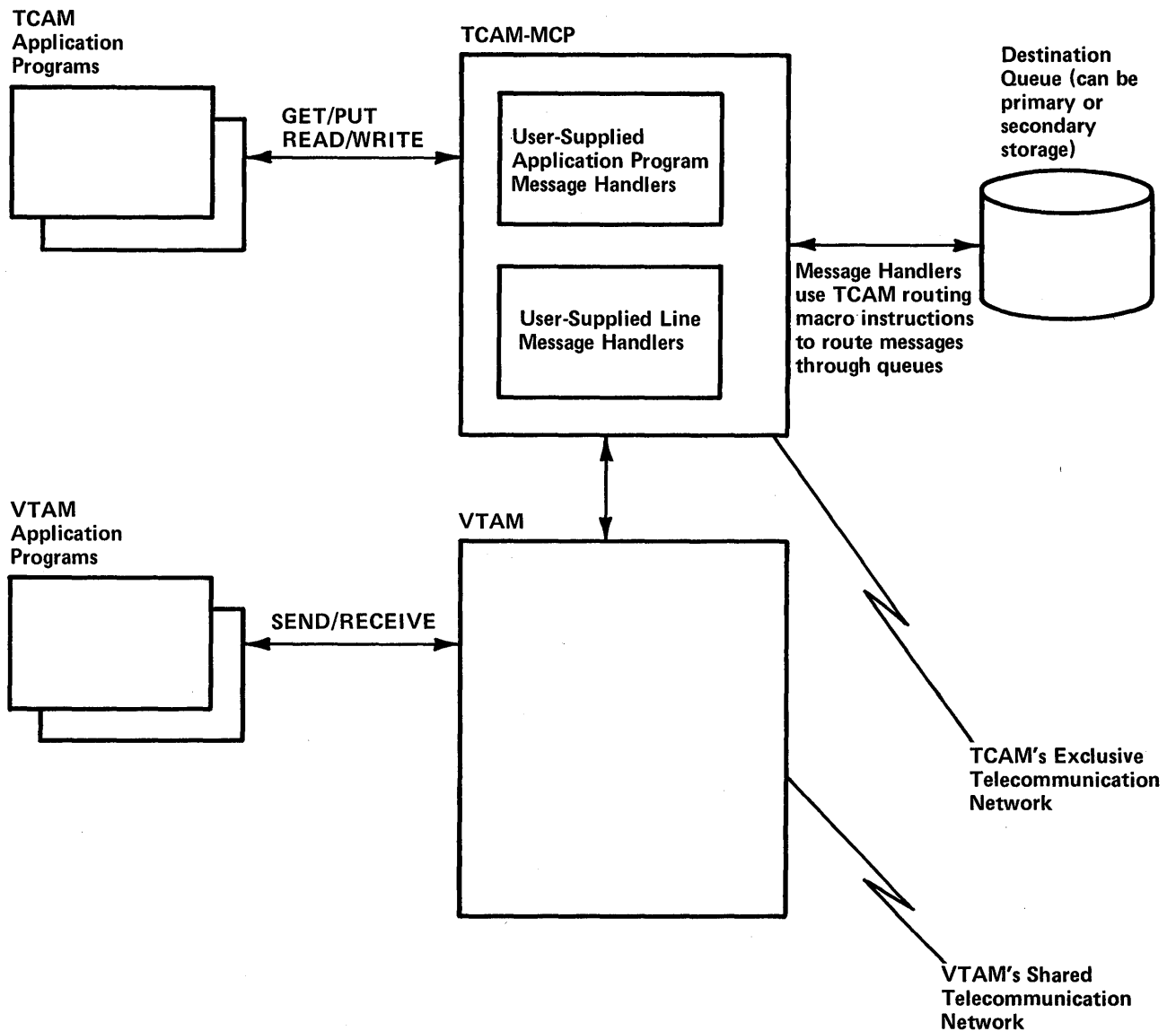


Figure 5-3. Relationship of TCAM to VTAM

application programs. The concepts and facilities described in this chapter all deal with these two aspects of an application program using VTAM—the application's ability to have a logical unit allocated to it, and its ability to communicate with that logical unit.

The facilities described first are those that apply generally to all of the application program's interactions with VTAM, whether they involve logical unit allocation (connection), or communication, or some request that is only indirectly related to allocation or communication. After a brief discussion of application program initialization, concepts and facilities relating to connection and to communication are discussed in detail.

### ***General Concepts and Facilities***

The application program issues macro instructions to request VTAM to perform some operation. VTAM communicates with the application program by setting register return codes, control blocks, and parameter lists, by posting ECBs, and by invoking special user-written routines. The manner in which VTAM communicates with the application program is to a considerable degree controlled by the application program. The following

Characteristic	VTAM Application Program	BTAM Application Program
<i>General Characteristics of the Access Method</i>		
Terminal-sharing	Yes	No
Line-sharing	Yes	No
Line-scheduling logic required	No	Yes
Polling required on part of application program	No	Yes
Exit-routines scheduled for special conditions such as a logon request	Yes	No
Local/remote 3270 handling	Can be communicated with in same mode as logical units; no coding distinction between local and remote	Requires different coding for local 3270 than for remote 3270
Primary type of terminal communication	Communicates primarily with logical units (program logic). Can also communicate with start-stop and BSC terminals	Communicates primarily with start-stop and BSC terminals
Direct-control or queued access method	Direct-control	Direct-control
<i>Language Characteristics</i>	VTAM macro instructions Keyword operands Macros common for DOS/VS and OS/VS Two sets of I/O macro instructions: record-mode (RECEIVE-SEND) for logical units and basic-mode (READ-WRITE) for BSC and start-stop devices Destination-oriented	BTAM macro instructions Positional operands Macros different for DOS/VS and OS/VS One set of I/O macro instructions Line-oriented
<i>Program Organization</i>	Telecommunication normally separate from processing Can have VTAM post an ECB or schedule an exit-routine when I/O event completes	Telecommunication normally separate from processing BTAM posts an ECB when I/O event completes

Figure 5-4 (Part 1 of 2). Major Similarities and Differences Between VTAM and BTAM Application Programs

facilities deal with the various ways that VTAM can be directed to handle the application program's requests and ways that the application program can be notified that particular events in the network have occurred.

**Overlapping VTAM Requests with Other Processing**

Each VTAM request issued by the application program can be handled *synchronously* or *asynchronously* by VTAM.

Synchronous request handling means that VTAM returns control to the next sequential instruction only after the requested operation has completed. Program execution is halted until VTAM determines that the operation has been completed. This type of request

Characteristics	VTAM Application Program	BTAM Application Program
<i>Functions Provided</i>		
Define line groups		Use DCB or DTFBT macro instruction
Define terminals	All resources are defined during VTAM definition	Use DFTRMLST macro instruction
Initialize program	Use OPEN macro instruction	Use OPEN macro instruction
Connect logical units dynamically to application program	Use OPNDST macro instruction	Not available, because terminals are statically connected when application's job step is initiated
Release control of a logical unit to another program that requests connection to it	When requested, use CLSDST with OPTCD=RELEASE specified	No comparable function
Receive input	Use RECEIVE macro instruction	Use READ macro instruction
from any connected logical unit/device	Specify input can be from any logical unit or device	Must poll each line separately
from a specific logical unit/device	Specify input is to be received from a specific logical unit or device	Specify terminal list entry
Send output	Use SEND macro instruction	Use WRITE macro instruction
Have data scheduled for output from access method buffers (output buffering)	Specify that the data is to be scheduled for output	No comparable function
Test, display, or modify control block fields	Use manipulative macro instructions: TESTCB, SHOWCB, MODCB. Or use assembler language instructions	Use assembler language instructions
Translate between EBCDIC and a transmission code	Not required for logical units. Performed in NCP for BSC and start-stop devices	Use TRNSLATE macro instruction
Record transmission errors	Performed by NCP and VTAM	Use error recording macro instructions

Figure 5-4 (Part 2 of 2). Major Similarities and Differences Between VTAM and BTAM Application Programs

handling is appropriate for applications that cannot continue processing until a particular request has completed. Figure 5-5 illustrates synchronous request handling.

Asynchronous request handling means that VTAM returns control to the next sequential instruction as soon as VTAM has accepted the request, not when the requested operation has been completed. Accepting a request consists of screening the request for errors and scheduling the parts of VTAM that will eventually carry out the requested operation. While the operation is being completed, the application program is free to initiate other I/O transactions or processing. For example, an application program might issue a

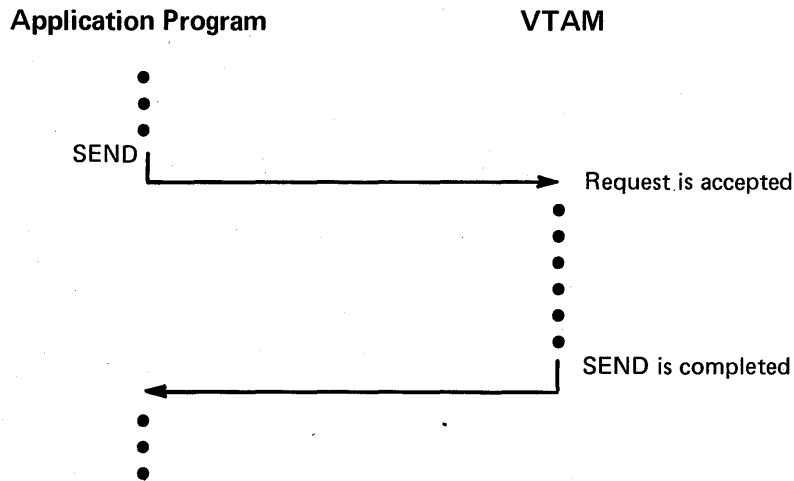


Figure 5-5. Processing Pattern for a Synchronous Request

RECEIVE macro instruction and indicate that it is to be handled asynchronously; while the input operation is being completed, the application program could begin to write to a direct-access storage device or communicate with another logical unit.

When asynchronous request handling is used, there are two ways that VTAM can notify the application program that the requested operation has completed. If the application associates an event control block (ECB) with the request, VTAM posts the ECB when the operation is completed. The application program can use a CHECK or WAIT macro instruction to determine that the ECB has been posted. Alternatively, the application program can associate an RPL exit-routine with the request. When the operation is completed, VTAM invokes the routine. Figure 5-6 illustrates asynchronous processing in an application program using ECBs; Figure 5-7 illustrates the use of the RPL exit-routine to control processing.

By using ECBs, the application program can use one WAIT macro instruction for a combination of VTAM requests and any non-VTAM requests that use ECBs. For example, an application program could issue three VSAM requests and three VTAM requests; by issuing one WAIT for all six ECBs, the application program can continue processing when any one of the six operations is completed.

ECBs also give the application program the freedom to determine during program execution when the program should stop processing and wait for a given operation to be completed. An application program might, for example, request data from a logical unit and then later determine that it should not stop and wait for that data (perhaps the application program has in the meantime begun to request data from another logical unit whose input is of a much higher priority and must be handled immediately). ECBs also allow the application program to “prioritize” requests by checking some ECBs before checking others.

The distinction between ECBs and RPL exit-routines rests primarily on the fact that the RPL exit-routine is *automatically* scheduled when the requested operation is completed, thereby saving the application program the trouble of checking ECBs and branching to subroutines. ECBs provide greater control, while RPL exit-routines provide greater convenience.

VTAM requests issued in the RPL exit-routine can also be handled asynchronously, although an exit-routine is not scheduled if another exit-routine has not completed. Figure 5-8 illustrates how an application program might use this facility.

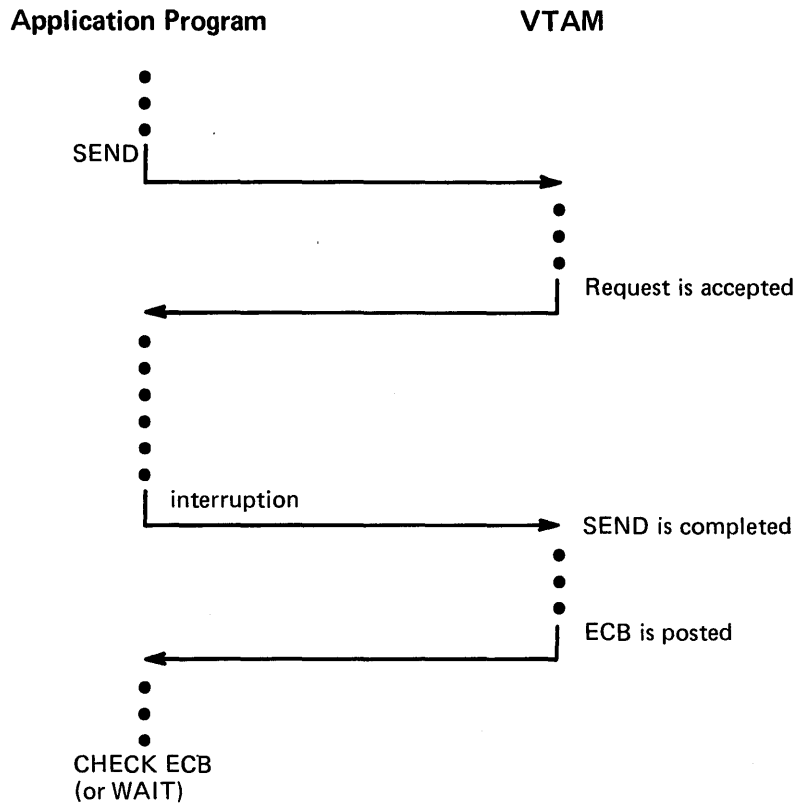


Figure 5-6. Processing Pattern When an ECB is Used with an Asynchronous Request

**Application Program  
Exit-Routines**

The application program can identify (via the EXLST macro instruction) a variety of exit-routines that VTAM schedules when particular events occur:

Event	Routine
A logical unit is waiting to be connected to the application program	LOGON
One of the logical units has withdrawn (or been withdrawn) from the network	LOSTERM
One of the logical units is wanted by another application program	RELREQ
The network operator is shutting down the network	TPEND
A start-stop terminal has caused an attention interruption	ATTN
A special type of input has arrived in the CPU (the types of input are discussed later)	DFASY RESP SCIP

When one of these events occurs, the execution of the application program is interrupted and the appropriate exit-routine is given control. If another event occurs while the exit-routine is still processing, the next exit-routine is not invoked until the first has completed (this applies as well for RPL exit-routines).



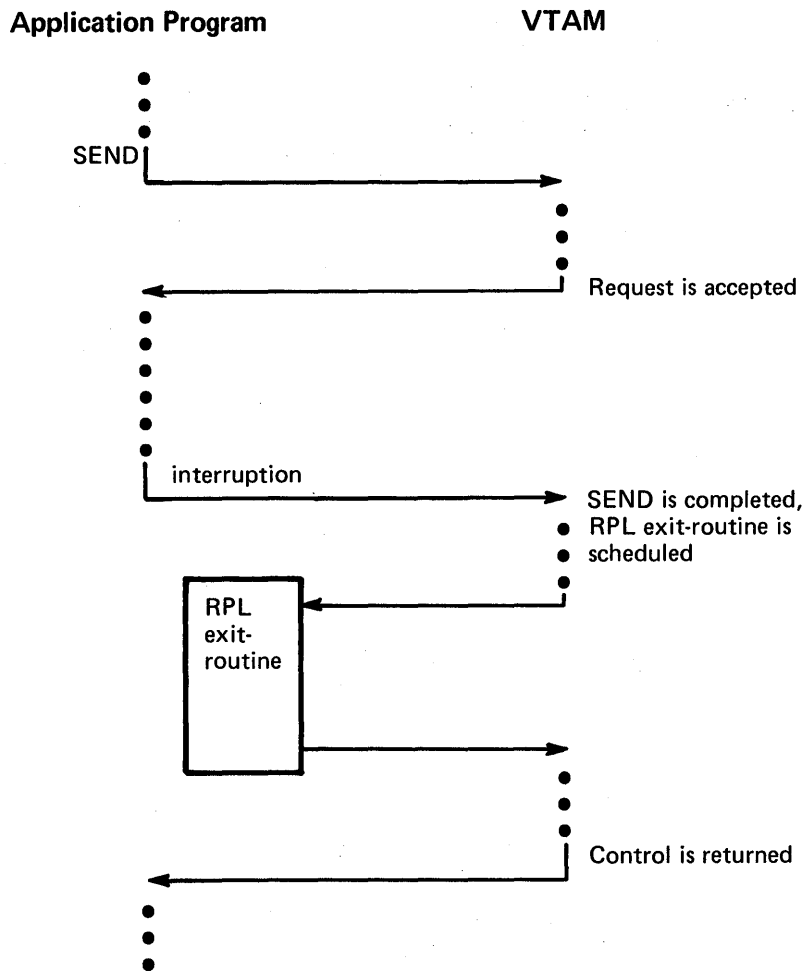


Figure 5-7. Processing Pattern When an RPL Exit-Routine is Used with an Asynchronous Request

Unlike the installation exit-routines (discussed in Chapter 3), which are included as part of the system during VTAM definition, the application program exit-routines are included as part of the application program. The addresses of these routines are placed in an EXLST control block by the application program.

Exit-routines other than the LERAD and SYNAD exit-routines need be reenterable only if two or more application programs share the same exit-routine.

In OS/VS, each exit-routine is usually scheduled under an interruption request block (IRB); in DOS/VS, each exit-routine is scheduled by changing the program information block (PIB) save area address. Any processing in the routine that places the routine in a wait state should be used with caution, since the application program's entire task waits while the exit-routine waits.

#### Error Notification

When synchronous request handling is used, error conditions are reported when the request has been completed and control returned to the application program. When asynchronous request handling is used, error conditions are reported in two stages. When control is first returned to the application program, VTAM indicates whether it has

## Application Program

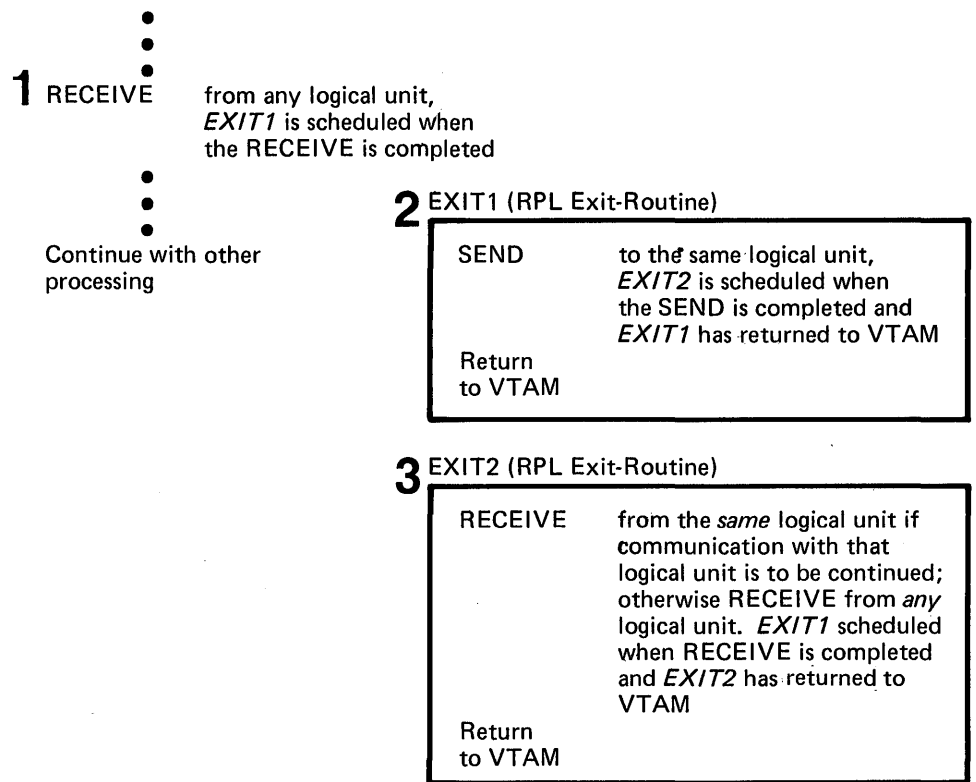


Figure 5-8. A Possible Processing Pattern When Asynchronous Requests Are Issued in RPL Exit-Routines

accepted or rejected the request. When an accepted operation completes, VTAM posts an ECB or schedules a designated RPL exit-routine, and, on the program's issuing a CHECK macro instruction, returns information about the completion of the requested operation. Figure 5-9 shows how errors are reported during asynchronous processing.

Success-or-failure information is presented to the VTAM application program in register 15 and, under some circumstances, in register 0. In addition, if the request or operation was not successful, VTAM will normally have placed additional information in return code fields of the RPL and will have attempted to enter the user's SYNAD or LERAD exit-routine.

The application program can code two error-handling routines that VTAM attempts to invoke as a result of all physical, environmental, and logical errors. The routine that handles logical errors is called the LERAD routine, and the routine that handles other errors detected by VTAM is called the SYNAD routine.

Should an error occur during synchronous request handling, the LERAD or SYNAD routine is invoked as part of the processing of that request. For an accepted asynchronous request, the routine is invoked when the request is checked by the application program (with a CHECK macro instruction).

When the LERAD or SYNAD routine receives control, it is provided in register 0 and in the RPL with information regarding the specific cause of the error.

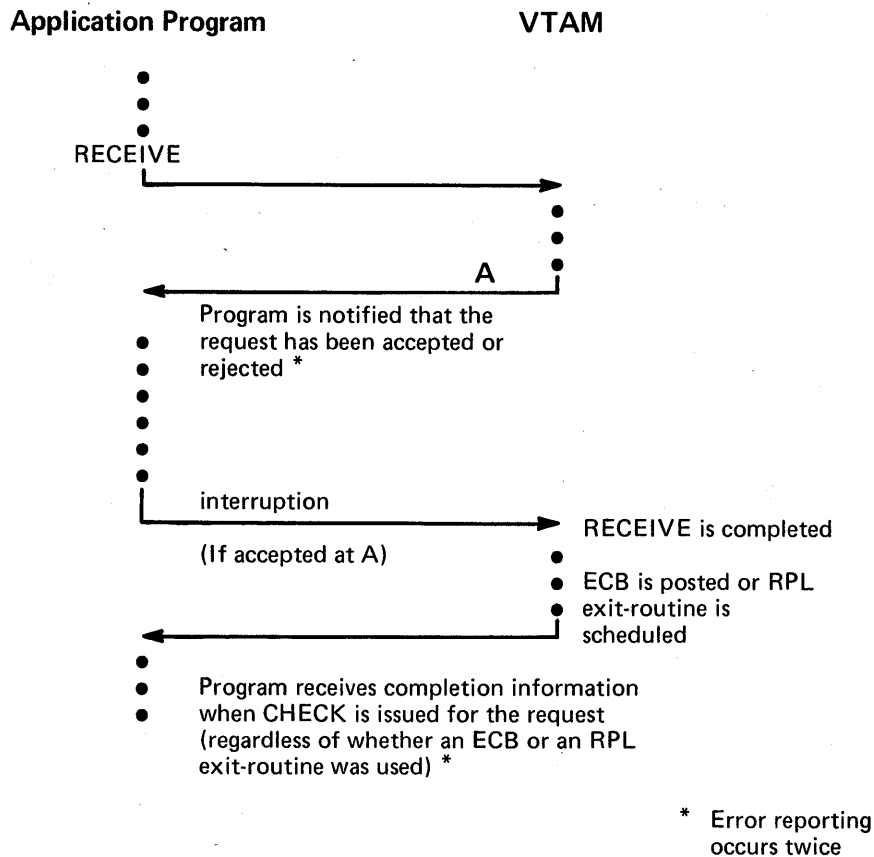


Figure 5-9. Processing Pattern For Reporting Errors During an Asynchronous Operation

The VTAM application program in the main part of the program or in a LERAD or SYNAD exit-routine can associate a class of completion information with a particular action. VTAM organizes its setting of register 0 and the RTNCD field of the RPL into these completion categories:

- Extraordinary completion. This requires further analysis of the RPL to determine the course of action required.
- Retryable completion. This indicates the request can be reissued. The EXECRPL macro instruction can be used.
- Damage completion. This indicates that, in addition to reissuing a request, some input or output data may have to be recovered or corrected.
- Environment error completion. This indicates that the program should call for external intervention.
- User logic error completion. This indicates that the program status should be saved, perhaps by requesting a dump. Depending on the seriousness of the error, the program can continue or terminate.

In some cases, simple determination that the return code (in register 0 and the RTNCD field of the RPL) specifies one of these completion categories can determine the course of action. In other cases, additional analysis of the RPL and other information (such as program flags) may be required to determine action.

**LERAD Processing:** Since most logical errors result from flaws in the design of an application program, the handling of logical errors is most important during the

development and debugging of the application program. The handling of the error may consist of little more than recording the attributes of the request that failed, requesting a dump, and causing an abnormal termination.

**SYNAD Processing:** Physical and environmental errors usually require more extensive treatment. The application program should, during program execution, determine the nature of the error, assess the extent of the problem, and attempt remedial action. If the application program determines that the error occurred because incoming data exceeded the capacity of the NCP buffers, for example, the application program could inform the logical unit that the data should be resent in two transmissions. If the problem is a recurring hardware check for the logical unit, however, the application program may have to take whatever action is required to continue without the logical unit.

The error-handling routine can set register 0 or register 15 and return via VTAM to the instruction following the synchronous request or CHECK macro instruction. If the exit-routine was able to handle the situation successfully, register 15 and, in some cases, register 0 can be set to zero to indicate that the requested operation completed normally. The main part of the program continues normally, unaware that an error or special condition occurred and that the LERAD or SYNAD exit-routine was entered. If, however, the exit-routine was not able to successfully complete the operation, it sets an understood value in one or both registers 15 and 0 before returning, so that the main program can take any action that the LERAD or SYNAD exit-routine did not take.

*Note:* The LERAD and SYNAD routines should be coded reenterable (1) if synchronous request handling or CHECK macros are issued in these routines, or (2) if synchronous request handling is used both in the main program and in any of the other routines (routines other than the LERAD or SYNAD routine).

### ***Opening the Application Program***

In other telecommunications access methods such as BTAM, the application program first associates itself with the access method and with the line groups needed to support communication. The VTAM application program also must establish an association with the access method. This is accomplished by issuing an OPEN macro instruction.

A VTAM application program, like a BTAM program, typically begins with an OPEN macro instruction and ends with a CLOSE macro instruction. That is, the association between the application program and VTAM normally lasts for the duration of the application program's execution.

VTAM's OPEN, however, does not associate the application program with any logical units (the VTAM application program communicates directly with logical units and is not aware of line groups). Logical units are associated with the application program by the process of connection.

The application program also uses the OPEN macro instruction to supply VTAM with the addresses of its error-handling routines (LERAD and SYNAD) and its event-driven exit-routines (such as LOGON, LOSTERM, and RELREQ).

### ***Connection***

An application program must establish connection with a logical unit before communication with that logical unit can begin. Connection is established with an OPNDST macro instruction.

The OPNDST (open destination) macro instruction causes VTAM to "allocate" the logical unit to the application program. OPNDST initializes VTAM's and the application's

control blocks to indicate that the logical unit and the application are connected. OPNDST also ensures that an active path exists between the two nodes before connecting them. Unlike the effect of an OPEN macro instruction, the effect of an OPNDST often does not last for the duration of the application program's execution. Because of VTAM's terminal-sharing facility, connections to the network's logical units can be made, broken, and remade innumerable times throughout the duration of the application program.

An application program can establish connection in one of two ways: it can *accept* the logical unit or it can *acquire* the logical unit.

## Acceptance

When an application program accepts a logical unit, it does so because a logon request was received for the logical unit. A logical unit cannot be accepted unless (or until) a logon request has been issued for it. Although there are several possible sources of the logon request—the network operator, another application program, the logical unit itself or automatically as a result of VTAM definition—the request usually originates *outside* of the application program. (Logon requests can also be generated *within* the application program; however such logon requests are essentially a form of acquisition, and are discussed in “Acquisition” later in this chapter.)

Acceptance is suitable for applications that do not require access to a specific logical unit or specific set of logical units in order to function, but instead are designed to service various logical units that require access to the application. If the installation, for example, wants the logical units themselves to designate which application they wish to use, the installation could allow each logical unit to initiate logon requests so that the application could accept the logical unit.

**Note:** When VTAM notifies an application program of a logon request, the logical unit and its logon request is “queued” to the application. As long as the logical unit is queued to the application program, it is not available for connection to other applications; it is available for connection only to the application to which it is queued. The application program and its queued logical unit are unable to communicate with each other until the connection is completed by the application's acceptance of the logical unit. Because a queued logical unit is effectively eliminated from the system until accepted or disconnected by the application, the installation should ensure that application programs avoid leaving logical units on this queue any longer than necessary.

**Accepting Logical Units with an Exit-Routine:** The application program can maintain a LOGON exit-routine which VTAM schedules whenever a logon request for the application program is made. VTAM provides the exit-routine with the identity of the logical unit associated with the logon request. The application program can either accept the logical unit (using an OPNDST macro instruction) or reject it (using a CLSDST macro instruction).

The application program does not have to use an exit routine in order to determine when a logon request has been made. The application program can instead issue a connection request that VTAM completes as soon as a logon request has been made. Although this method is simpler to use than an exit-routine, the application program does not have the opportunity to decline the logon request prior to actually establishing connection with the logical unit. (A logon request is declined by issuing a CLSDST macro instruction.)

**Preventing Logon Requests:** Logon requests cannot be directed at an application program until the application program, using the SETLON macro instruction, notifies VTAM that it is ready to accept them. At this point, any logon requests that are directed at the application program are queued until the application program either accepts or rejects them. Any time during its execution, the application program can notify VTAM that it is no longer accepting logon requests.

**Types of Acceptance:** The application program can issue a connection request to accept a *specific* logical unit, or to accept *any* logical unit for which a logon request has been issued.

To accept a specific logical unit, the application program must tell VTAM the identity of the logical unit; connection is not made until a logon request has been issued for that particular logical unit. This is the type of acceptance that an application program would use in the LOGON exit-routine. Outside of the exit-routine, the application program can accept a logon request from any logical unit in the network. After connection is established, VTAM passes the identity of the logical unit to the application program.

## Acquisition

When the initiative for connection originates from the application program, the application program establishes connection by acquiring the logical unit. No logon request need exist; if the logical unit is active and not being used by another application program, the logical unit is connected (or queued for connection if the application is simulating a logon request to itself on behalf of the logical unit). The installation must authorize the application program's use of acquisition.

**Types of Acquisition:** Some or all of a set of logical units can be acquired at once. As many as are available are connected. This type of connection (called the CONALL option) can be used when the application program is willing to proceed with as many logical units as are available. VTAM provides information so that the program can determine which logical units were connected.

Another type of acquisition allows the application program to acquire any *one* logical unit of a specified set. The first available logical unit of the set is connected. This type of acquisition (called the CONANY option) is useful for applications that require a logical unit, but for which one logical unit is as good as another.

The application program specifies the logical unit set by building a series of control blocks (NIBs) each containing the installation-supplied name of the logical unit. The set can be limited to one logical unit.

A third type of acquisition can be used by application programs that both accept and acquire connections or which wish to ask the current owner of a logical unit (through its RELREQ exit-routine) to release it. An application program can itself generate a logon request for a logical unit, and let the part of the program that accepts logon requests establish connection with the logical unit. The application program can generate logon requests for an entire set of logical units, or for any one of a set of logical units.

The use of such logon requests, called simulated logon requests, is a form of acquisition since the initiative for connection lies with the application program. Like the other forms of acquisition, its use must be authorized by the installation. Simulated logon requests are generated with the SIMLOGON macro instruction.

**Acquiring Connected Logical Units:** One application program cannot take another application program's logical unit from it. If an application program attempts to acquire an already-connected logical unit, no reconnection is possible until the current owner of that logical unit disconnects it. VTAM provides a means by which the owning application program can be notified that another application program is requesting one of its logical units.

The requesting application program can indicate whether its attempt to re-acquire a connected logical unit should or should not cause the owning application program to be notified. The requesting application program should request this notification when it

needs the logical unit regardless of its connection status. Notification should not be indicated when the requesting application program only needs the logical unit if it is unconnected.

The owning application program also controls whether it can be notified when another application program issues a connection request to acquire one of its logical units. Notification can therefore only occur when both application programs so indicate.

VTAM notifies the owning application program by scheduling the RELREQ exit-routine which the application program maintains for this purpose.

The RELREQ exit-routine is provided with the identity of the contested logical unit. The application program can elect to disconnect the logical unit immediately, disconnect it later, or ignore the request entirely. If the logical unit is disconnected, the previous owner can immediately attempt to re-acquire the logical unit from the new owner (using a queued connection request as described below) so that the logical unit will be returned when it is no longer being used. When the logical unit is disconnected, it is reconnected to the acquiring application program that has waited the longest—this may not necessarily be the application program that was the previous owner of the logical unit.

By controlling which application programs release contested logical units and which do not, the installation can cause some application programs to be able to obtain and keep logical units more readily than other application programs. Or, the installation could establish a policy that all application programs release contested logical units that are not being used; this would maximize the overall utilization of the logical units.

### Queuing Connection Requests

Application-program requests for connection always fail if the logical unit is inactive. If the logical unit is active but unavailable, however, the application program designates whether its connection request should fail or whether the request should remain pending (queued) until the logical unit does become available.

Although the definition of an “available” logical unit differs between acquired and accepted connections, the option of queuing or not queuing the communication request applies to both. When an application program attempts to *acquire* an active logical unit, the logical unit is available if it is not connected (or queued for connection as the result of a logon request) to another application program. When an application program attempts to *accept* a logical unit, the logical unit is available if a logon request for it has been directed at the application program. (Note the distinction between queuing an application program’s request for connection—described here—and queuing a logical unit to an application program as the result of a logon request—as described in “Acceptance” earlier in this chapter.)

Figure 5-10 lists the effects of queuing a connection request on the various types of connection requests.

### Disconnection

An application program can disconnect a logical unit in one of two ways: it can *release* the logical unit or it can *pass* the logical unit to another application program. The logical unit is released by disconnecting it without regard to which application program (if any) is to receive the logical unit. The logical unit is passed by disconnecting it and designating which application program is to receive the logical unit. Passing must be authorized by the installation.

Passing and releasing is accomplished by using the PASS and RELEASE options of a CLSDST macro instruction.

Type of Connection Request	Meaning When Connection Is To Be Queued	Meaning When Connection Not To Be Queued
<p><i>Acquire</i></p> <p>as many as are available in a set</p> <p>a set of one</p> <p>any one of a set</p>	<p>Schedule a LOGON exit-routine as each logical unit becomes available.*</p> <p>Schedule a LOGON exit-routine when the logical unit becomes available.*</p> <p>Schedule a LOGON exit-routine for the first logical unit in the set that becomes available.*</p> <p>* Request queueing is available for SIMLOGON only. It is not available for OPNDST with OPTCD=ACQUIRE.</p>	<p>Connect all the logical units that are available</p> <p>Correct the logical unit if it is available</p> <p>Connect the first logical unit in the set that is available; otherwise connect none</p>
<p><i>Accept</i></p> <p>a specific logical unit</p> <p>any logical unit</p>	<p>Connect the logical unit if a logon request has been received from it; otherwise connect the logical unit when a logon request is received from it</p> <p>Connect any logical unit from which a logon request has been received (if more than one has been received, connect the logical unit that has waited the longest); otherwise wait for a logon request and then connect the logical unit</p>	<p>Connect the logical unit (only if a logon request has already been received from it)</p> <p>Connect any logical unit from which a logon request has already been received (if more than one has been received, connect the logical unit that has waited the longest); otherwise connect none</p>

Figure 5-10. Queued and Unqueued Connection Requests

If the logical unit is released, VTAM connects the logical unit to any application program that has attempted to acquire the logical unit (and has indicated that its connection request should be queued). If more than one application program has attempted to acquire the logical unit, VTAM connects the logical unit to the application program that first issued the connection request. If there are no queued requests to acquire the logical unit, VTAM generates an automatic logon request for the logical unit; if no automatic logon request has been specified by the installation, the logical unit remains unconnected.

When a logical unit is passed, VTAM disconnects the logical unit, generates a logon request, and directs the logon request to the designated application program. The logical unit is not reconnected until the receiving application program accepts the logon request.

A logical unit should be passed only when it is imperative that it be connected to a specific application program and to none other. For example, an installation might maintain several application programs that each require the same information from the logical unit before any of them can be used. Although each application program could conduct its own interrogation, it might be simpler for the installation to maintain a single application program that converses with the logical unit to obtain the initial information and then passes the logical unit to the appropriate destination.

When the application program passes a logical unit, it can also pass data to the receiving application program. In the example above, the application program might pass along the results of the preliminary conversation.



## Communication

Communication between application programs and logical units differs in many ways from communication with start-stop and BSC terminals. Most of the differences result mainly from the programmable nature of logical units.

In other terminal-oriented access methods such as BTAM, the application program communicates with terminals whose physical characteristics are known. In VTAM, the application program is communicating with a type of terminal (logical unit) whose *logical* characteristics must be known.

If, for example, the program in the logical unit performs all device-dependent I/O operations, the application program using VTAM would not be device-dependent; it could send data to the logical unit without regard to the device-control and formatting characters needed to display the data. On the other hand, the logical unit might *not* perform such device-dependent operations; it would then be the application program's responsibility to do so.

The designer of the application program must also be aware of how the program in the logical unit handles the various communication facilities described below.

As the remaining aspects of communication are read, the following concept should be kept in mind: The application program is communicating with other user-written programs, rather than with terminals having fixed physical characteristics.

## Messages and Responses

Communication between an application program and a logical unit consists of an exchange of *messages* and *responses*. Messages consist of data records and/or control information (called indicators). The length of data records is defined by the application program and the logical unit. Data records represent whatever amount of data the application program or the logical unit chooses to send at one time. Responses are acknowledgments that messages have been received; responses contain response information and indicators.

Indicators help VTAM and the connected application program and logical unit control communication between the two nodes. Indicators are discussed throughout this section and are summarized in Appendix D.

Messages and responses are sent by both the application program and the logical unit. That is, the application program sends messages to the logical unit and receives responses from it; the logical unit sends messages to the application program and receives messages from it. The relationship between messages and responses is shown in Figure 5-11.

A node that is expecting a response but has not yet received it can send a *chase indicator* to the other node. A response to a chase indicator indicates that all responses have been sent.

Messages can be sent to a logical unit with one of two options:

- The application program can indicate that as soon as the message has been scheduled for transmission and the output data area is free, VTAM is to consider the output operation completed (by returning control, posting an ECB, or scheduling an RPL exit-routine). This is called *scheduled* output and is illustrated in Figure 5-12.
- The application program can indicate that VTAM is not to consider the operation complete until the message has been received by the logical unit and a response has been returned. This is called *responded* output and is illustrated in Figure 5-13.

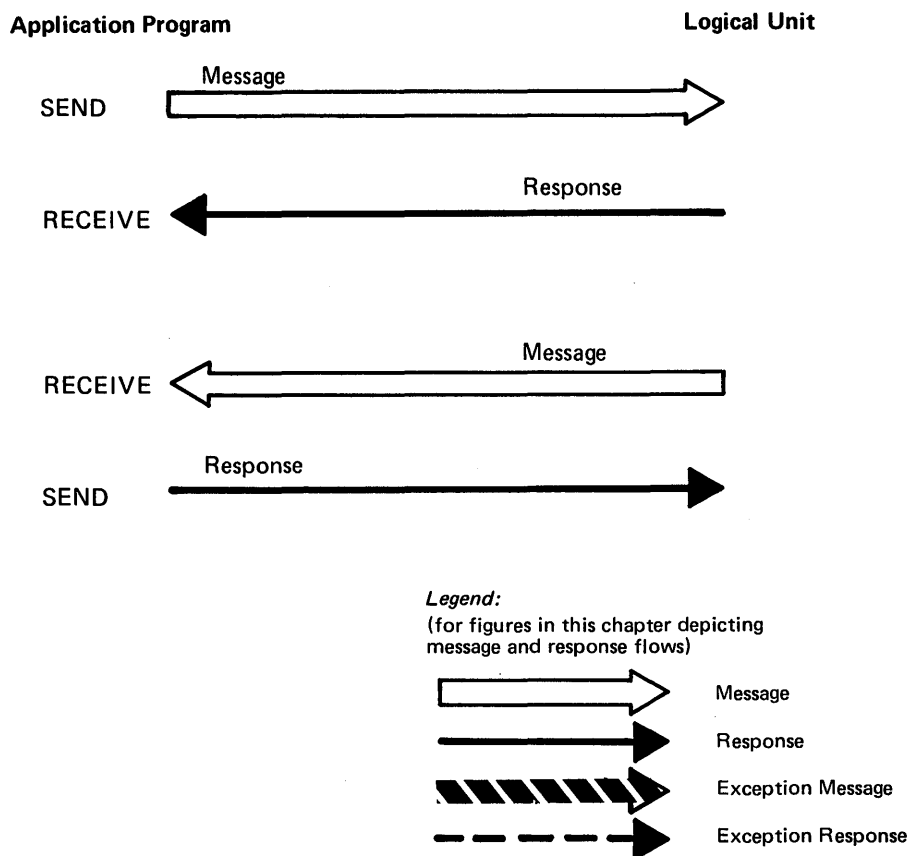


Figure 5-11. Exchanging Messages and Responses

These options define what constitutes the completion of an output operation, and should not be confused with synchronous and asynchronous request handling, which indicate the action to be taken when the completion occurs.

Responded output is the easiest to use, but requires that the output data area and RPL not be reused until a response has been received. If the response indicates that an error occurred, the data is still available for retransmission. Scheduled output allows the application program to send a series of messages that all use the same I/O area and RPL.

With responded output, completion status information is returned when the output request is completed. But with scheduled output, the output request is completed before any completion status information is available. To determine how the output operation completed, the application program must issue an input request to obtain the completion status information. This is why the application program in Figure 5-12 issues three input requests in addition to the three output requests.

If an error occurs during the transmission of the message, the receiver is passed a substitute message that indicates the error condition. This message is called an *exception message*. The node transmitting the message becomes aware of the error condition when the other node returns an *exception response*.

When the logical unit sends a message to the application program, the logical unit indicates on that message the type of response that it expects in return. The terminal can "tell" the application program to :

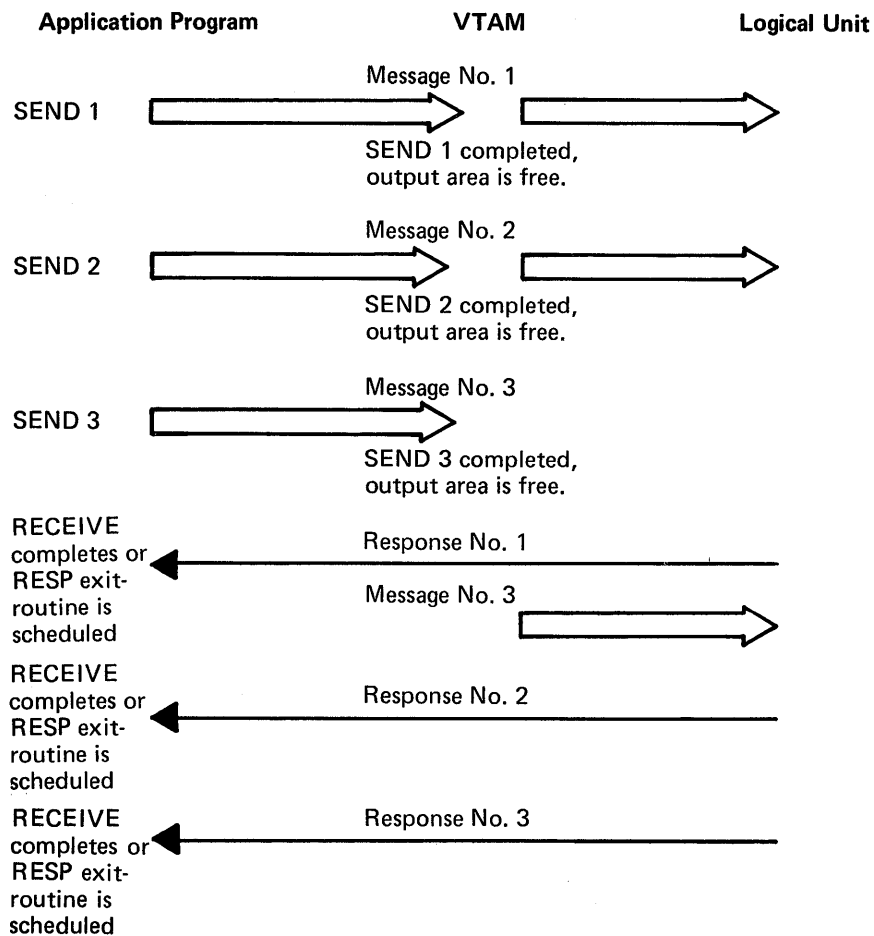


Figure 5-12. Scheduled Output

- Send a response regardless of whether the message arrives normally or not. That is, if an exception message is received, send back an exception response. If the message arrives normally, send back a normal response. Part A of Figure 5-14 is an example of responding to both normal and exception messages.
- Send only an exception response. That is, if the message arrives normally, send no response. If an exception message arrives, send an exception response. Part B of Figure 5-15 is an example of responding only to exception messages.
- Send no response at all.

Requesting both normal and exception responses, rather than just exception responses, results in greater control over error conditions and provides an opportunity for quicker error recovery—but requires more programming. A request for only exception responses can be used within a group of related output messages if the entire group is to be discarded when an exception condition occurs. Requesting no responses of either sort is appropriate when there is no concern for error recovery for that transmission.

The application program can specify the same options in the messages that it sends to a logical unit. The logical unit can examine the message received from the application program and determine whether an exception or normal response (or no response) should be returned to the application program.

The logical unit or application program can send an exception response to a message that was successfully received for reasons other than whether or not it was received

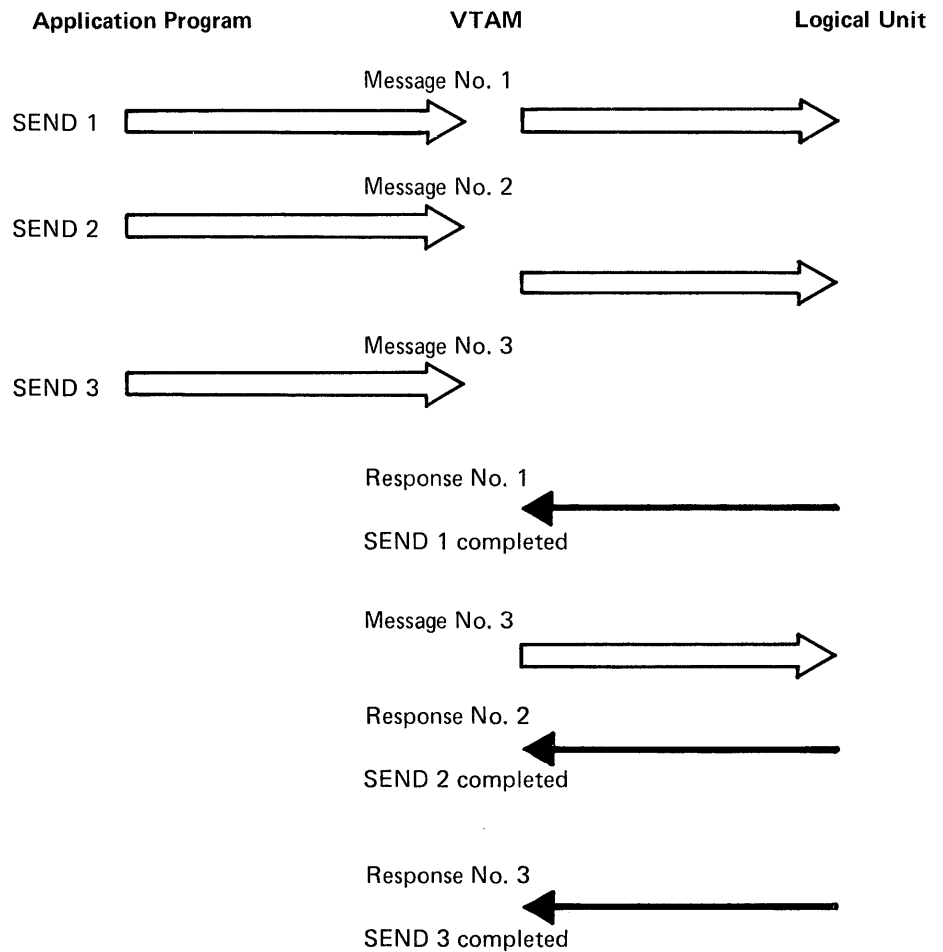


Figure 5-13. Responded Output

successfully. For example, an exception response might indicate that the data in the message was not in a prescribed form.

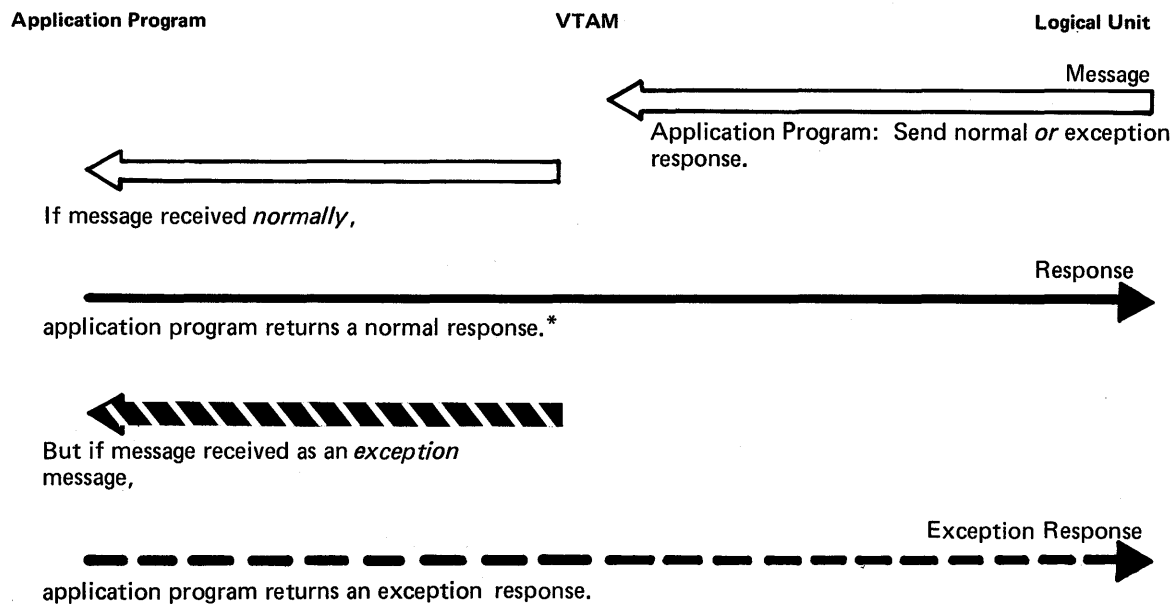
### RRN and FME Responses

Every response, independent of its normal or exception aspect, is designated by its sender as an *RRN response* or as an *FME response*. RRN responses are used to control data recovery; FME responses are used to indicate the arrival of data at its ultimate destination.

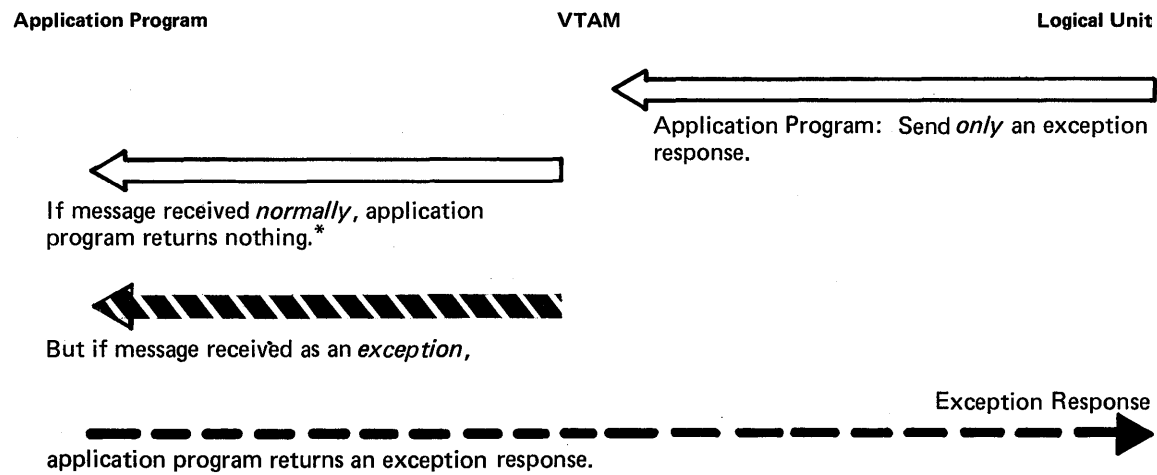
A normal RRN response signifies that the response sender (1) has successfully received the data, and (2) accepts recovery responsibility for the data. Recovery responsibility means that the node will hold a copy of the data until the data has successfully reached its destination. An exception RRN response signifies that the response sender either has not successfully received the data, or that it cannot accept recovery responsibility for the data.

A normal FME response, on the other hand, signifies that the data has reached its final destination. A 3600 work station, for example, might return a normal FME response to indicate that it has successfully written the data to one of its output terminals. An exception FME response signifies that the data did not successfully reach its final destination.

## A Logical Unit Requests Either Normal or Exception Response



## B Logical Unit Requests Only Exception Response

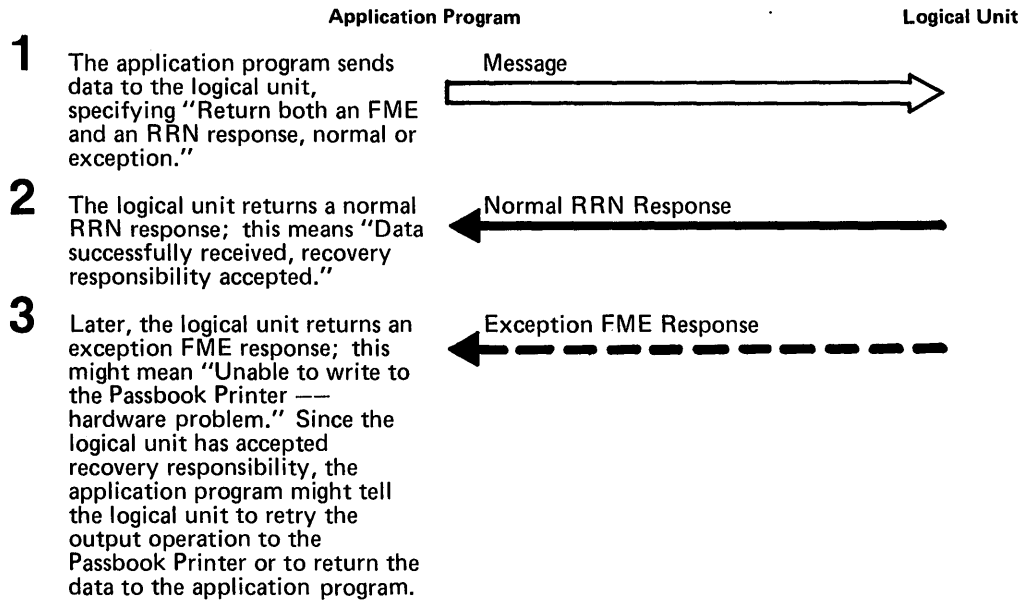


\* Even though a message arrives normally (as far as VTAM is concerned), the application program can determine for its own reasons that the message is "defective" and return an exception response, rather than a normal or no response.

Figure 5-14. Two VTAM Responses to Messages

Both the application program and the logical unit specify for each *message* the expected types of responses, and specify for each *response* the type of response being sent

### A Request Originating in Application Program



### B Request Originating in Logical Unit

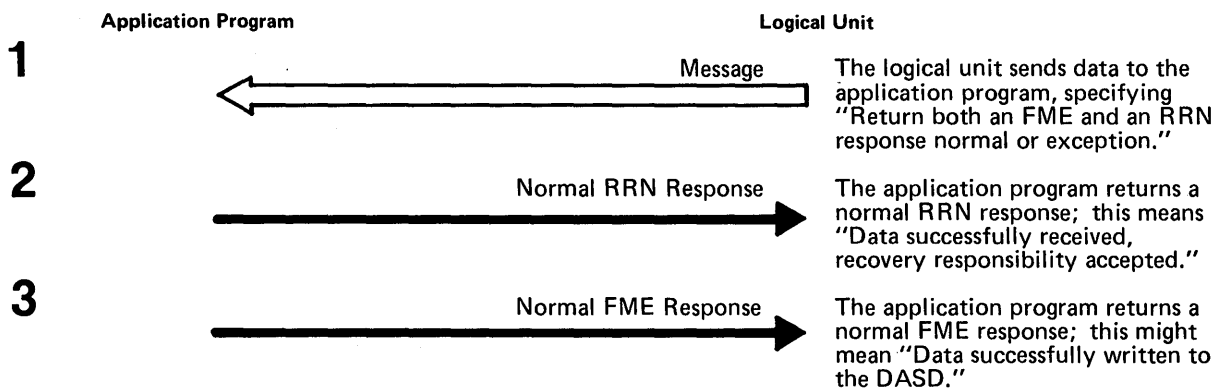


Figure 5-15. Two Possible Uses of The Response Types

The application program indicates on each message whether it expects an FME response, an RRN response, or both to be returned. Combining these types of responses with the normal/exception response types described above yields seven possible combinations of response types that can be indicated for a given message:

- Return a normal or exception FME response.
- Return a normal or exception RRN response.
- Return both an FME and an RRN response, normal or exception.
- Return only an exception FME response.
- Return only an exception RRN response.
- Return both an FME and an RRN response, exception only.
- Return no responses of any sort.

The logical unit, like the application program, also specifies for each message the types of expected responses. Figure 5-15 illustrates two uses of FME and RRN responses.

### Sequencing and Chaining

Each data record sent to a logical unit is assigned a sequence number by VTAM. The numbering begins with the first record sent after connection. The number is increased by one for each subsequent record. This process continues until the logical unit is disconnected, unless interrupted earlier by the application program.

Should a message arrive out of sequence (that is, its sequence number is not 1 greater than that of the last record received), VTAM considers this to be a transmission error and replaces the missing message with an exception message.

When a response is sent (either normal or exception), the sender assigns to it the sequence number of the message being responded to. This provides the node that originally sent the message with a means of matching the response with its message. For example, an application program might send a group of messages, with each message indicating that only exception responses should be returned. Should an exception response be returned, the application program could use the sequence number to determine where in the group the error occurred. Sequence numbers are also useful for logical units that log each received or sent message.

Application programs (or logical units) can group any number of messages into a set called a message *chain*. The sender can indicate which part of a chain is being transmitted—the first message of the chain, the last message of the chain, neither (the message is somewhere in the middle) or both (the message is the sole element of the chain).

The node transmitting a chain can at any time send a *cancel indicator* to the receiving node (the node might send this indicator because an exception response has been returned). The receiving node can interpret this indicator as an indication that all received records of the current chain should be discarded.

The actual unit of work that the chain represents is determined entirely by the application program and the logical unit.

Figure 5-16 illustrates a possible use of chaining. In this example, a logical unit has submitted an inquiry to the application program. The application program can obtain various pieces of information from data files and send them to the logical unit as each becomes available. By chaining the output requests, the application program has a convenient way of telling the logical unit whether any given piece of data represents the beginning, middle, or end of a reply to an inquiry and of checking all the reply before displaying any part of it.

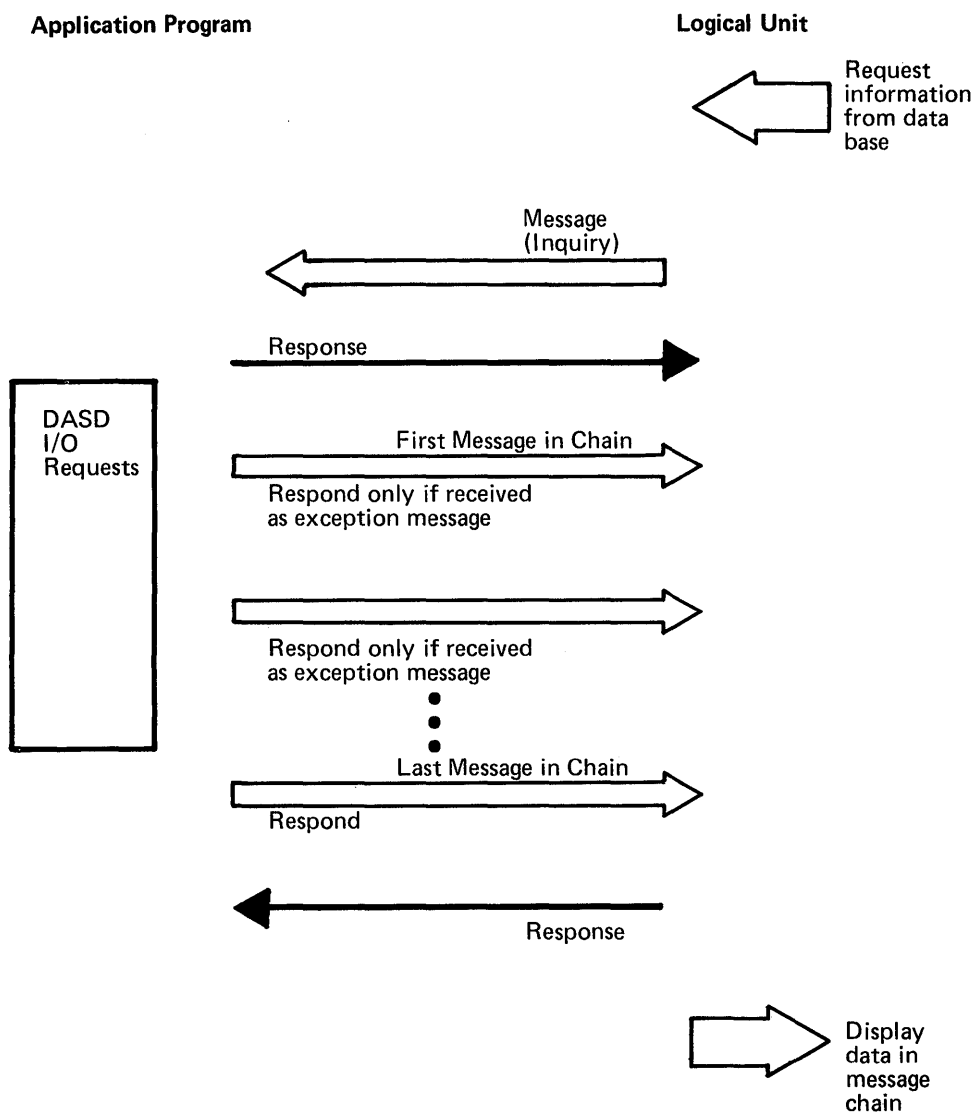


Figure 5-16. An Example of Message Chaining

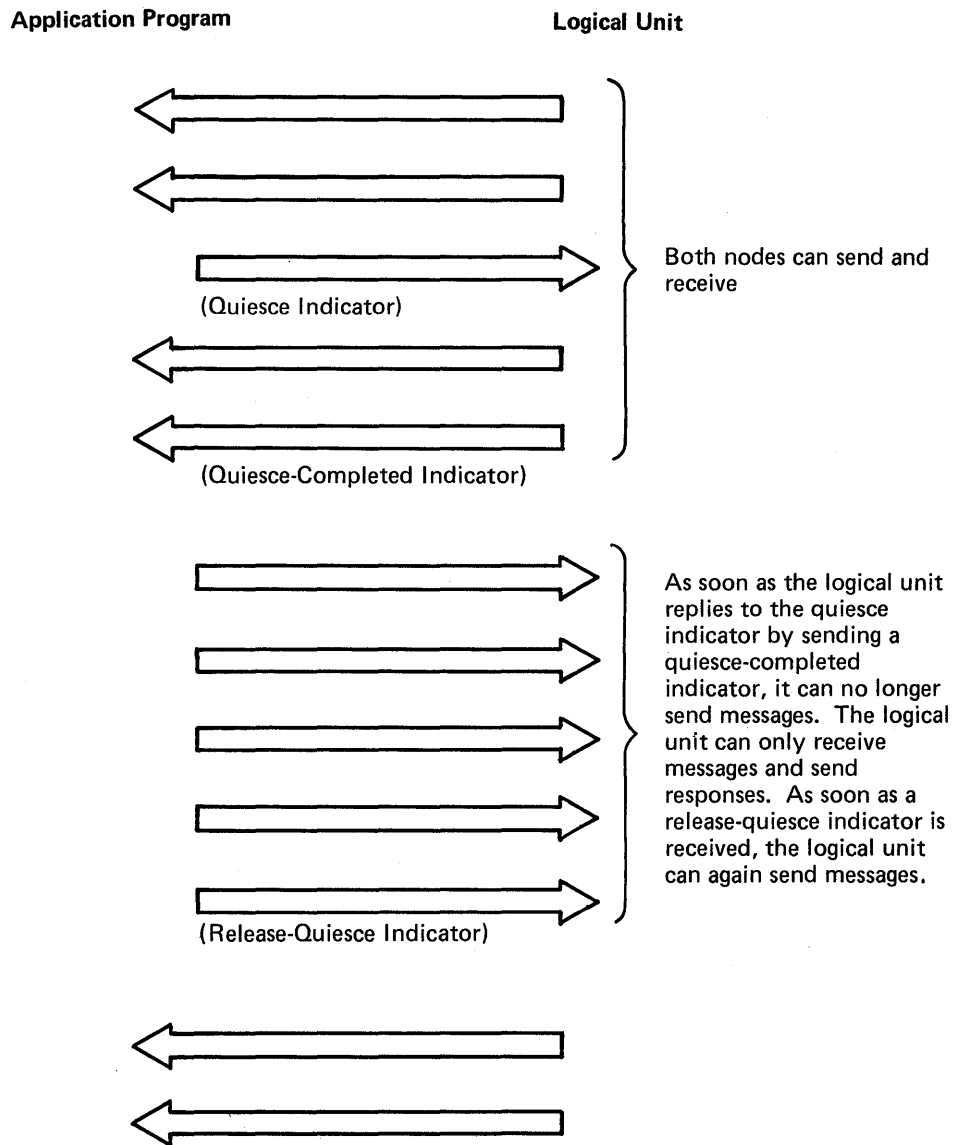
## Quiescing

VTAM provides a set of indicators that the application program can use to request a logical unit to stop sending synchronous-flow messages to the program. The VTAM application program can receive similar requests from a logical unit for the application to stop sending.

Quiescing can be used to occasionally stop the logical unit or application program from sending because of some temporary circumstance. For example, a logical unit may be receiving a long chain being sent from a VTAM application program. Elements being received are stored in a buffer associated with the logical unit. When a certain point in the buffer is reached (a point that normally is not reached), the logical unit sends a *quiesce-at-end-of-chain (QEC) indicator*. The VTAM application program can interpret this indicator to mean "quiesce immediately." It can either end the chain at this point, allowing the logical unit to clear its buffer, or it can send back a quiesce-completed indicator, indicating that it is temporarily ceasing to send until the logical unit clears its buffer and sends back a *release-quiesce (RELQ) indicator*.

In another example of the use of quiescing, a terminal operator at a device to which a printout was being sent could notify the subsystem application program to interrupt the





Note: Responses are not shown

Figure 5-17. An Example of Quiesce Communication

printout to enter an inquiry. The logical unit would send a QEC indicator, and the VTAM application program would temporarily hold sending and prepare to receive input from the logical unit.

### Facilities for Ensuring Orderly Communication

Messages can be sent to a logical unit without regard to whether the logical unit is at that moment sending messages to the application program. The nature of some applications, however, may prohibit such unrestricted exchange of data, or the application program may have been developed when no capability for unrestricted data exchange existed, and the installation does not wish to rewrite the program just to use this facility. For such application programs, three methods of communication are available that allow the application program and the logical unit to control each other's ability to send data. These three methods (which are essentially three sets of indicators with "rules" about how to use them), are called *quiesce communication*, *change-direction communication*, and *bracket communication*.

**Quiesce Communication:** The application program informs the logical unit that it is to stop sending data when the logical unit has completed sending its current chain. It does so by sending a *quiesce-at-end-of-chain* (QEC) *indicator*. When the logical unit replies, by sending a *quiesce-completed* (QC) *indicator*, it must stop sending and prepare to receive. The logical unit cannot again send data until it receives a *release-quiesce* (RELQ) *indicator* from the application program, as shown in Figure 5-17.

This convention is not enforced by VTAM; it is the user's responsibility to conform to the convention.

The application program is not necessarily the primary node (that is, the node entitled to "quiesce" the other node).

**Change Direction Communication:** The first node that sends data (following the application program's indication that data flow can begin) can continue to send data. When the first node is through sending data, it sends a *change-direction-command indicator* to the other node. The other node sends data until it relinquishes its ability to send by returning another change-direction-command indicator. The nodes continue to alternate in this fashion, as shown in Figure 5-18.

While the receiver is awaiting the change-direction-command indicator, it can transmit (as part of a response) a prompting indicator to the other node that in effect says "I would like that change-direction-command indicator now." The prompting indicator, called a *change-direction-request indicator* can be honored or it can be ignored.

Change-direction is not enforced by VTAM. Should the node waiting for a change-direction indicator begin sending data anyway, VTAM does not prevent the transmission of the data. The successful use of this method of communication rests on the assumption that neither the application program nor the logical unit violates the "rules."

The node that is awaiting a change-direction indicator, like the node that is in a quiesced state, is free to send responses. Only the sending of messages is restricted.

**Bracket Communication:** A *bracket* is any unit of work that the application program and the logical unit have been programmed to accomplish. Each bracket consists of input operations or output operations (or both) that do not necessarily follow a fixed pattern. Data-base inquiry and data-base update transactions are typical examples of brackets.

Bracket communication is used when one of the nodes cannot process a new bracket until the previous one has been completed. Nodes using this method of communication note on each transmitted message whether that message is the beginning or end of a bracket. These delimiters allow the receiving node to determine whether or not a new bracket can be started. Figure 5-19 illustrates a use of bracket communication.

Because either connected node can initiate a bracket, a *bid indicator* can be used to avoid situations in which both nodes attempt to initiate a bracket at the same time. A bid indicator requests permission to start a bracket. Upon receipt of a bid indicator, the receiving node can reject it, give permission for a bracket to be initiated, or reject the bid temporarily. If the bid is rejected temporarily, the node that received it transmits a *ready-to-receive* (RTR) *indicator* when a bracket session can be permitted. Upon receipt of an RTR indicator, the node that originally sent the bid can initiate the bracket by sending a begin-bracket indicator in a message.

To use the bracket convention effectively, only one node should be permitted to initiate a bracket without the use of the bid. The other node should be required to use the bid before initiating a bracket.



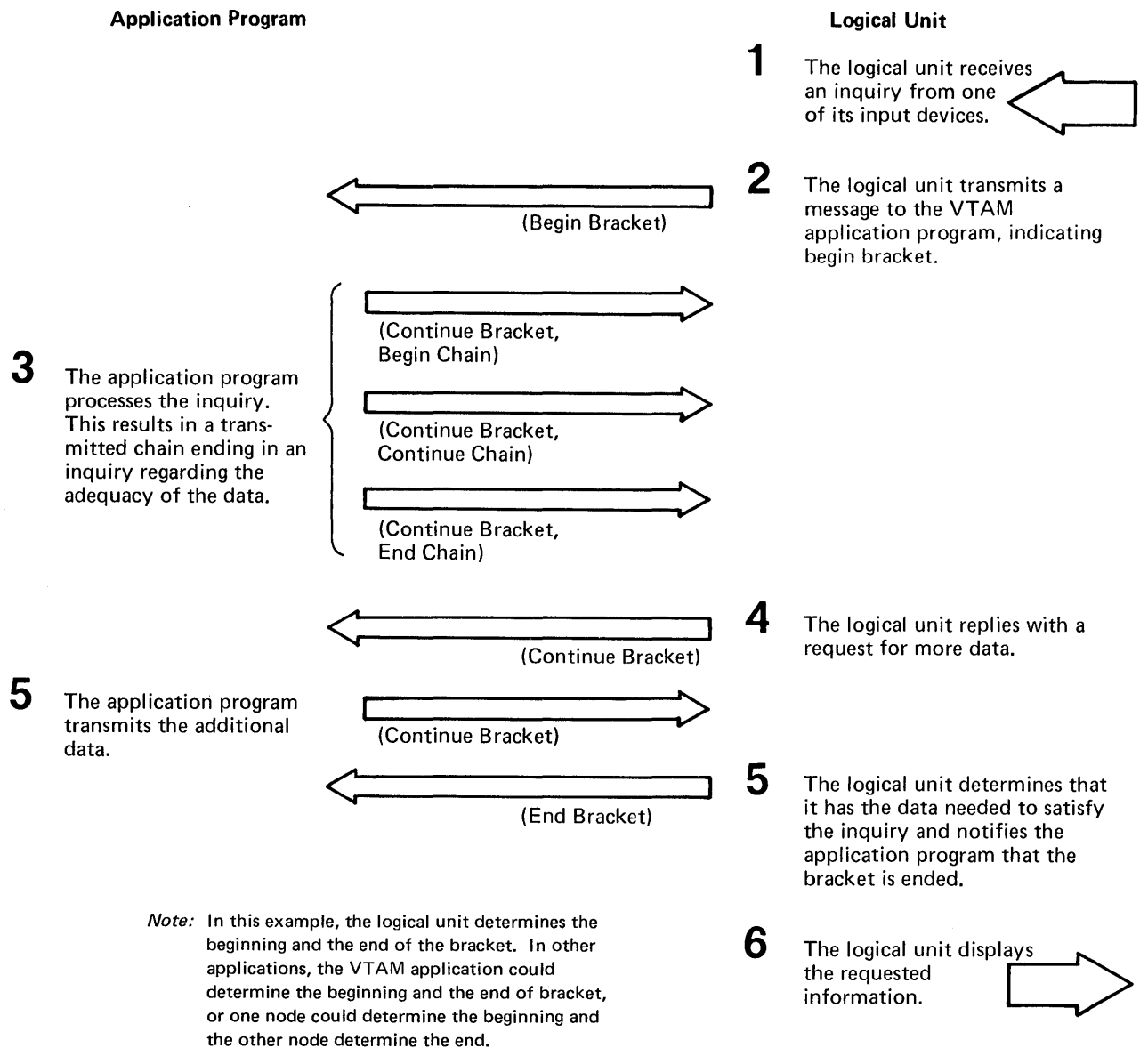


Figure 5-19. An Example of Bracket Communication

other are discarded, and further transmissions are prohibited. An SDT indicator is used to resume the flow of messages and responses. The flow of messages and responses can be stopped and restarted any number of times as illustrated in Figure 5-21.

The SDT and clear indicators are not sent in the same manner as messages and responses (that is, with the SEND macro instruction) but are sent with a SESSIONC macro instruction.

Another indicator sent by the SESSIONC macro instruction, called the *set-and-test-sequence-number (STSN) indicator*, allows the application program to reset the VTAM-incremented sequence number, or to communicate with a logical unit to establish the proper sequence number. For example, the application program can send a sequence number to the logical unit and from the response determine if the logical unit "agrees"

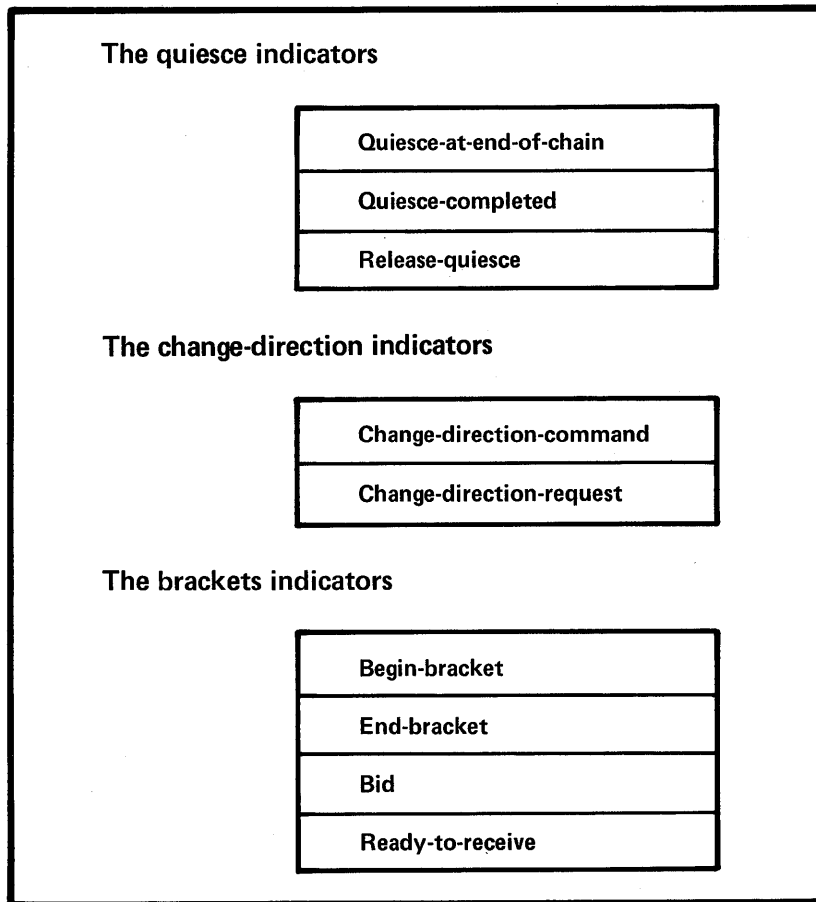


Figure 5-20. Indicators Used to Direct the Flow of Data

with that number. Or, the application program can simply “ask” the logical unit to return whatever value it considers to be the correct sequence number.

STSN indicators are used in conjunction with SDT and clear indicators in a process called *sequence number recovery*. This process begins when the application program or the logical unit determine that messages are arriving out of sequence. (If the logical unit initiates the process, it first sends the application program a *request-recovery (RQR) indicator*.) The application program either on its own initiative or as the result of receiving the request-recovery indicator, establishes the correct sequence number—typically in the following manner:

1. The application program sends a clear indicator to stop the flow of messages and responses. This is necessary to insure that no pending transmissions complete while the application program and the logical unit are in the midst of determining the correct sequence number.
2. The application program, by sending various STSN indicators and examining responses to them, communicates with the logical unit and establishes the correct sequence number.
3. The application program issues another STSN indicator to reset the VTAM-supplied sequence number to the new value.
4. The application program completes the recovery process by sending an SDT indicator which allows the flow to resume.

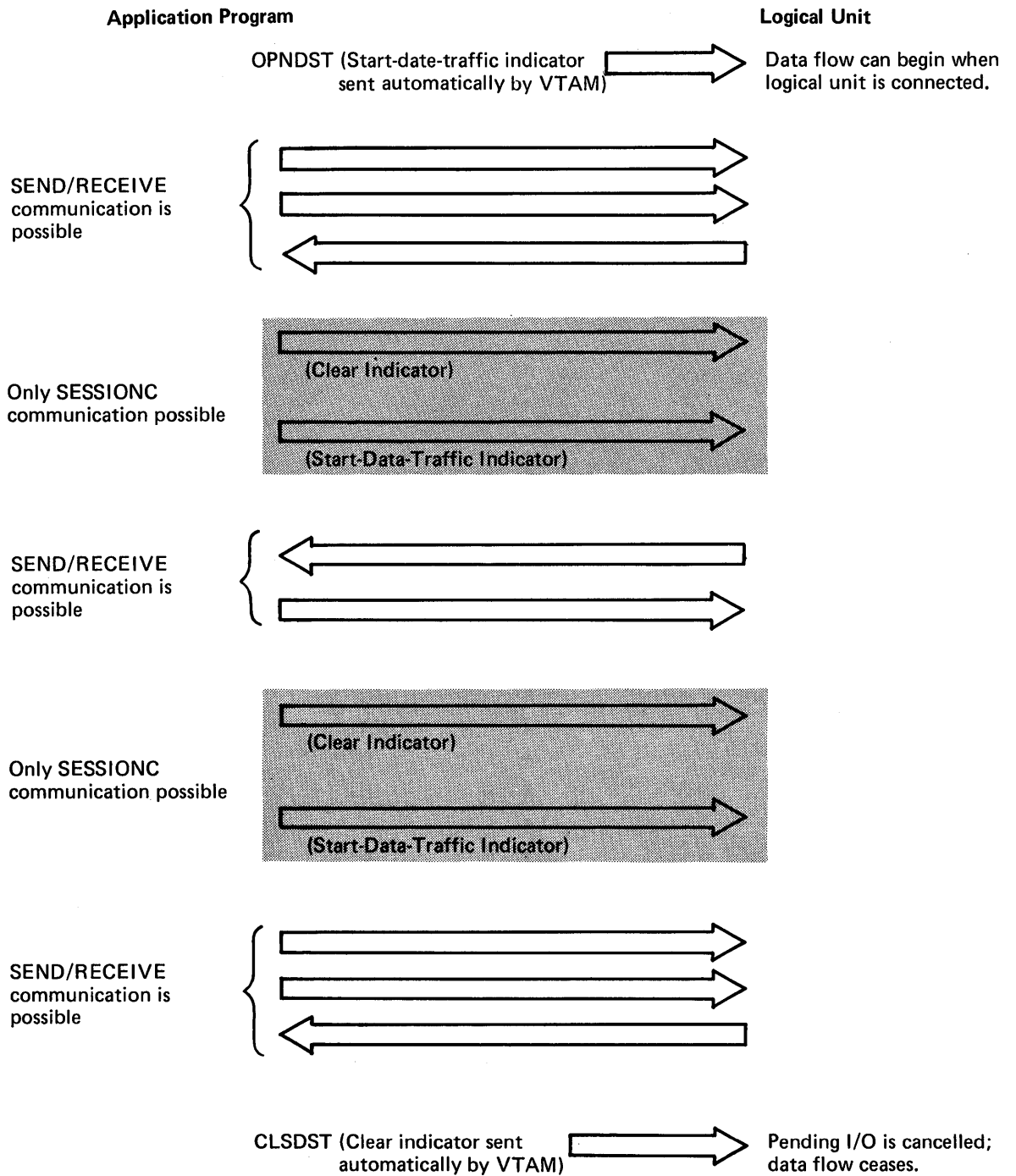


Figure 5-21. An Example of Start-Data-Traffic and Clear Indicators

## Receiving Input

Data records, responses, and data flow control information can be received by the application program separately or with one RECEIVE macro instruction. When the macro instruction is issued, it indicates which one of the following types of input can satisfy the request (any combination can be specified):

- Synchronous-flow messages.
- Asynchronous-flow messages.
- Responses.

Synchronous- and asynchronous-flow messages are terms used to group all messages into two distinct types of input. (Synchronous- and asynchronous-flow messages should not be confused with synchronous and asynchronous request handling. The two are unrelated.)

The need for the two types of input results from these characteristics of data transmission:

- If messages are sent at a faster rate than the other node receives them, the messages are queued for the receiving node.
- Some messages are more important than others, in the sense that they should not be queued with the other messages but should be available to the receiving node separately and immediately.

Thus, VTAM provides two “priority levels” for messages. Data records are always treated as synchronous-flow messages. Certain flow-control information is treated the same way. One example is the quiesce-completed indicator, which must keep its place in the queue of data records; if it were to be received prematurely, the bypassed data records might be lost.

Other data-control information must be made more immediately available to the receiver and is therefore made available to the receiver as asynchronous-flow messages. One example of an asynchronous-flow message is the QEC indicator. This type of indicator is not meant to stay within a queue of data records, waiting until the receiver eventually obtains it.

See Appendix D for an indication of which indicators are synchronous-flow and which are asynchronous-flow.

Three types of exit-routines can be maintained by the application program that VTAM schedules whenever one of the following types of input become available:

- Asynchronous-flow messages (DFASY exit-routine).
- Responses (RESP exit-routine).
- Request-recovery (RQR) indicators (SCIP exit-routine).

Note that with one exception, the types of input that can cause the exit-routine to be scheduled correspond to the type of input that causes a particular type of input request to be completed. The exception is the SESSIONC indicator. Unless a SCIP routine is maintained for this purpose, the application program has no means of receiving an RQR indicator.

These exit-routines operate in the same manner as those described in “Application Program Exit-Routines,” earlier in this chapter. One difference, however, is that the application program need not use one exit-routine to handle a particular kind of input from *all* logical units. A given exit-routine can service input from a limited set of logical units, or a separate exit-routine can be maintained for each logical unit.

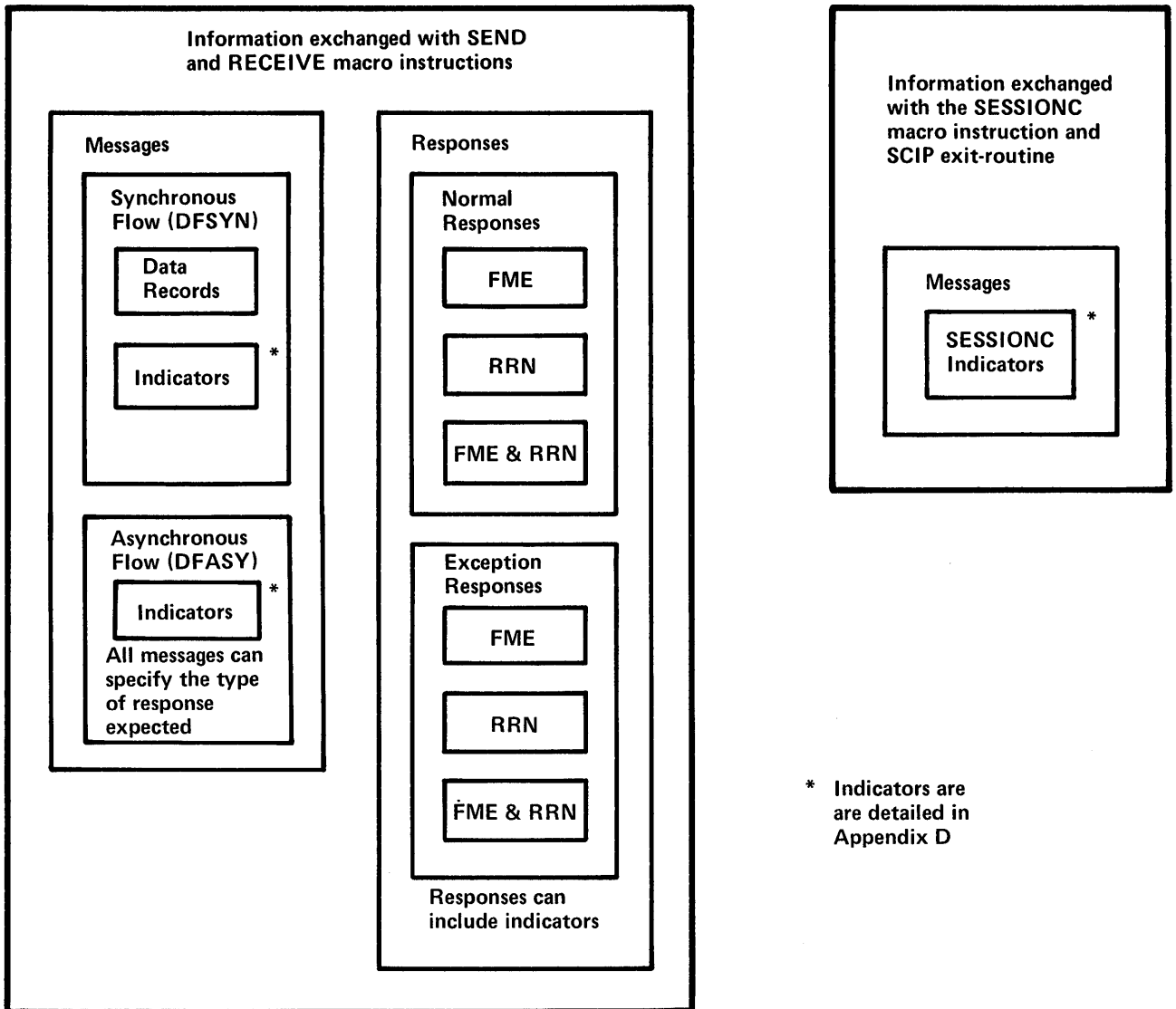


Figure 5-22. Types of Information Exchanged Between an Application Program and Logical Unit

Figure 5-22 summarizes the various messages and responses described above.

### Specific-Mode and Any-Mode

The application program can obtain data from its logical units in one of two ways; it can request data from a *specific* logical unit, or it can request data from *any* one of its connected logical units. The application program designates the desired mode—specific or any—with each RECEIVE macro instruction. These two modes are called, respectively, the *specific-mode* and the *any-mode*.

In general, an application program initially requests input from its logical units in the any-mode, and then proceeds to communicate with each logical unit in the specific-mode until the transaction, inquiry, or conversation is completed. While specific communication proceeds with one logical unit, the application program keeps a RECEIVE macro instruction (issued in the any-mode) pending so that new transactions can be handled.

A RECEIVE issued in the any-mode is analogous to a BTAM READ Initial macro instruction except that (1) the concept of polling does not apply to logical units, and (2)



the RECEIVE can be satisfied by any logical unit connected to the application program, rather than by just those attached via one line.

In the any-mode, the application program does not know the identity of the source logical unit until the data has been moved into its input area and the RECEIVE has been completed. Since the logical unit is initially unknown, the amount of incoming data may also be unknown. This means that the application program must either reserve an input area large enough to hold the largest possible amount of incoming data, or execute additional instructions to handle overlength data. On the other hand, the any-mode allows the application program to use just one input area for data from all of its logical units, rather than using a separate input area for each of its logical units.

With the specific-mode, the application program must specify the identity of the logical unit supplying the data. Since the identity of the source is known, the size of the input area is much more predictable than with the any-mode. A disadvantage is that since any given logical unit may not supply data for some time, the application program may have to contend with unused data areas.

Input data areas can be more efficiently managed by a combination of specific-mode and any-mode. As an example, consider an application program that obtains an inquiry from any of its logical units, handles that inquiry with a series of SEND and RECEIVE macro instructions, and then obtains a new inquiry. Part of such a program is illustrated in Figure 5-23.

#### Continue-Any and Continue-Specific Modes

In the example of Figure 5-23, synchronous request handling for the I/O requests is assumed. The application program handles each inquiry serially, never accepting a new inquiry until it has completed the previous one. Although this procedure might be suitable for application programs processing short inquiries and few logical units, most applications would probably work much better if the inquiries were handled in parallel.

An application program designed to handle more than one inquiry concurrently (Sample Program 2 at the end of this chapter is such a program) might use asynchronous request handling and issue new RECEIVE macro instructions in the any-mode before the previous inquiry has been completed. This, however, raises the possibility that both a RECEIVE for a specific logical unit and a RECEIVE for any logical unit (which includes the specific logical unit as well) might be awaiting data at the same time. Consequently, data that is meant to satisfy the subroutine's RECEIVE might instead be intercepted by the RECEIVE in the main program, which is meant only to receive new inquiries.

To eliminate this sort of problem, VTAM allows the application program to indicate when a particular logical unit's data can be received by a RECEIVE macro instruction issued in the any-mode, and when the data must be received by a RECEIVE macro instruction issued in the specific-mode. The former is called *continue-any mode*, and the latter is called the *continue-specific mode*. These modes are designated when an I/O request is issued, but do not become effective until the I/O operation is completed.

Figure 5-24 illustrates how the various modes described above relate to one another.

#### Identifying Logical Units

Before an application program begins to communicate with a logical unit, it has available to it the logical unit's installation-supplied name. This is an 8-byte symbolic name created for the logical unit during VTAM definition. When connection is established with the logical unit during program execution, the application program is also provided with a VTAM-supplied network-oriented name (called a communications identifier, or CID) for that logical unit. The application program uses the CID for all I/O requests issued in the specific-mode.

## Application Program

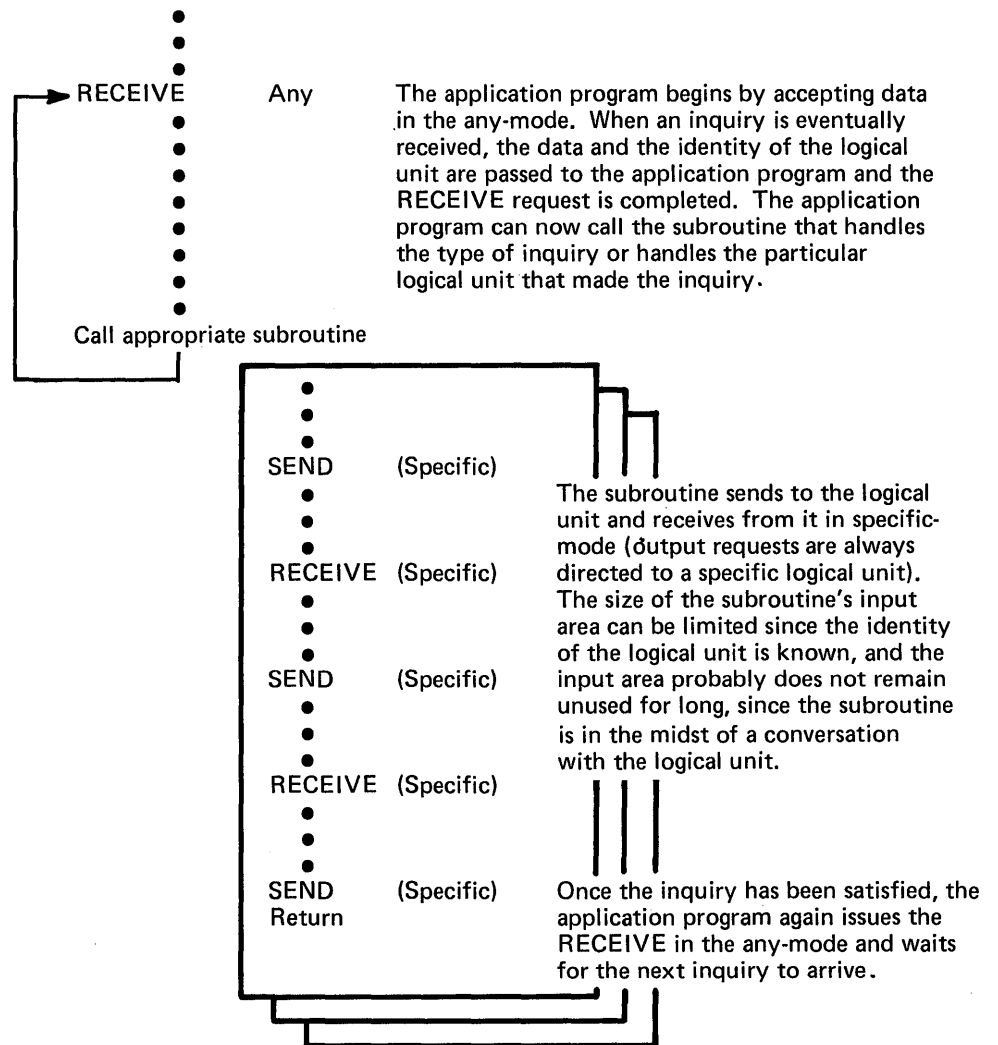


Figure 5-23. Using a Combination of Any-Mode and Specific-Mode to Obtain Data

When a RECEIVE macro instruction issued in the any-mode is completed, VTAM provides the identity of the logical unit that sent the data. Since the application program will probably proceed to communicate with the logical unit in the specific-mode, it is the CID, rather than the symbolic name, which VTAM supplies to the application program. Should the identity be significant, the application program has three ways to relate the CID to the logical unit's symbolic (installation-supplied) name:

- The application program can ask VTAM to translate the CID into its symbolic name. This is done with an INQUIRE macro instruction.
- The application program can maintain a table of CIDs and their symbolic equivalents.
- When the application program establishes connection with the logical unit, the application program can initially assign to the logical unit a four-byte value that VTAM returns each time that logical unit's data satisfies a RECEIVE. The four-byte value can be anything the application program chooses to associate with the logical unit and is known as the user field. It could be used to identify the logical unit, or it could contain the address of a subroutine that is to handle that logical unit's data.

## Application Program

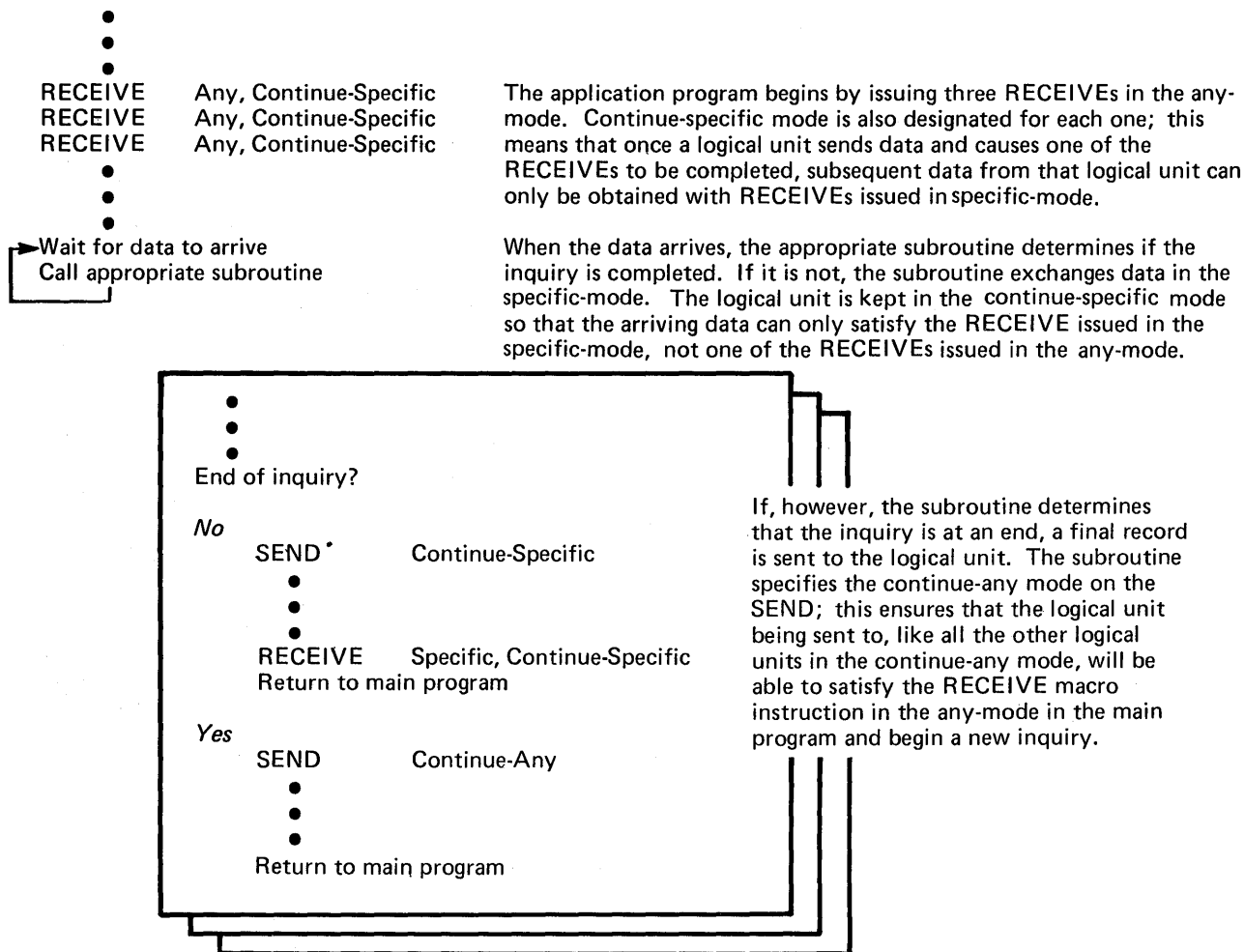


Figure 5-24. Using The Continue-Any and Continue-Specific Modes to Handle Concurrent Inquiries

### Handling Overlength Input Data

When an application program issues a RECEIVE macro instruction, the length of the incoming data is often unpredictable. As noted above, this is particularly true of RECEIVE macro instructions issued in the any-mode. VTAM provides two ways of handling data that is too large for the input area:

- VTAM can discard the overlength data. This facility is called the TRUNC (truncate) option.

This option would be useful in applications that must impose rigid size limitations on input data. For example, an inventory-control application might require the logical unit to supply an account number no longer than 10 bytes.

- VTAM can keep the data. VTAM fills the input area, saves the remainder, and completes the input request. Additional input requests must be issued to obtain the excess data. This facility is called the KEEP option.

The application program selects the appropriate option on an individual logical unit basis when the logical unit is connected and can override it on a per-request basis.

## Communicating with the 3270 Information Display System

Although the 3270 is not a logical unit, the application program communicates with each 3270 as though it were. That is, the application program uses SEND and RECEIVE macro instructions in much the same manner as described above.

VTAM provides the SEND-RECEIVE facility so that application programs communicating with logical units can use the same macro instructions to communicate with 3270s. This facility also allows a program that communicates with logical units to be tested with 3270 terminals.

*Note:* The application program has the option of communicating with the 3270 terminals in the same manner used to communicate with BSC and start-stop terminals. See “Communicating with Start-Stop and BSC Terminals,” below.

Since the 3270 is not a logical unit and has no programmable capabilities, VTAM cannot make a 3270 appear exactly like a logical unit to the application program. Consequently, the following restrictions apply to all SEND and RECEIVE communication with a 3270 (all other aspects of communication apply as described previously):

- All commands and orders for the 3270 must be placed in the output data by the application program. Data, therefore, includes 3270 commands and orders.
- No responses should be sent to a 3270. All incoming messages indicate that no response of any type is expected.
- Messages sent to a 3270 can contain only data and 3270 commands and orders. No quiesce, change-direction, bid, chase, or cancel indicators should be sent. Bracket indicators can be set, but chaining indicators should always mark the message as the sole element of a chain (all incoming messages are so marked).
- No RQR indicators can be received from a 3270. Only the clear indicator can be sent to it. The effect of the clear indicator is to reset both incoming and outgoing sequence numbers to 0.
- The bracket convention must be used. If the application program has no use for brackets, the entire interval between the first I/O operation of a connection and disconnection can be considered to be one bracket. Both the application program and the 3270 can begin a bracket.

The first input from a 3270 that begins an NCP session is marked as the beginning of a bracket. All subsequent messages received from the 3270 during the NCP session indicate that the bracket is being continued. The 3270 cannot end a bracket; this can only be done by the application program.

- The application program should request FME responses for each message sent to the 3270 that begins or ends in a bracket.

Those readers concerned only with how VTAM is used to communicate with logical units and 3270 terminals should skip the following part and resume reading at “The VTAM Language.”

## Communicating with Start-Stop and BSC Terminals

The macro instructions and facilities used by the application program to communicate with VTAM-supported start-stop and BSC terminals are different from those used to communicate with logical units. The two sets of macro instructions and facilities are called the *basic-mode* and the *record-mode*. The remainder of this section discusses basic-mode.

The basic-mode must be used for the start-stop and BSC terminals, and may also be used for the locally and remotely attached 3270. The record-mode is used for all logical units and for the locally and remotely attached 3270. Appendix B lists the logical units, BSC terminals, and start-stop terminals supported by VTAM.

The record-mode macro instructions are the SEND, RECEIVE, RESETSR, and SESSIONC macros mentioned above. The basic-mode macro instructions are these:

#### SOLICIT

Poll or prepare a terminal (or all connected terminals) for input, obtain a pre-established amount of data from the terminal, and move the data into VTAM buffers.

#### READ

Move data from VTAM buffers into the application program's input area. A variation of this macro instruction includes the actions performed by the SOLICIT macro.

#### WRITE

Write a block of data to a terminal.

#### RESET

Cancel a pending SOLICIT, READ, or WRITE operation for a terminal, and/or reset the NCP error lock for the terminal.

#### DO

Execute a set of logical device orders (LDOs) to perform particular operations.

Many of the concepts and facilities that have been discussed in this chapter apply to start-stop and BSC communication as well as to communication with logical units. The following concepts and facilities are common to both the basic-mode and the record-mode:

- Connection.
- Overlapping VTAM requests with other processing.
- Application program exit-routines. (However, the DFASY, SCIP, and RESP exit-routines apply only to logical units.)
- Error notification.
- Specific-mode and any-mode.
- Continue-any and continue-specific modes.
- Terminal identification. (See "Identifying Logical Units," earlier in this chapter.)
- Handling overlength input data.

The following concepts and facilities are those which are used only for basic-mode communication.

**Data Blocks:** The unit of data exchanged between the application program and a terminal depends on whether record-mode or basic-mode is being used. Messages (which include data records) and responses are exchanged in record-mode. In basic-mode, the unit of data is the *block*.

Blocks are delimited differently for different types of terminals. For start-stop terminals, a block ends with an EOB character; for BSC terminals, a block ends with an ETB or ETX character.

Although the application program can solicit more than a block from a terminal, a READ macro instruction can move only a block into the application program's input area (or less, if the input area is smaller than a block). Output operations (WRITE macro instructions) always send one block to the terminal.

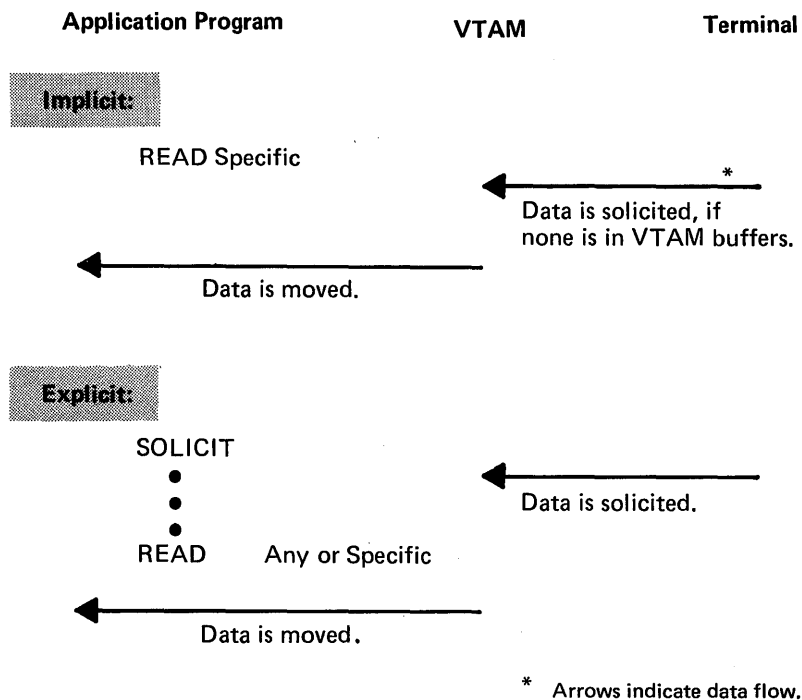


Figure 5-25. Implicit and Explicit Solicitation in the Basic-Mode

**Solicitation:** When the application program solicits data from a terminal, VTAM initiates whatever actions (such as polling or line preparation) are required to obtain data from the terminal into VTAM buffers.

READ requests issued in the specific-mode cause solicitation to take place if no previously-solicited data is in VTAM's buffers, and then cause the data to be moved into the application program's I/O storage area. This is similar to a READ Initial macro instruction in BTAM. In contrast, READ requests issued in the any-mode can only move solicited data from VTAM's buffers into the application program's input area. This is similar to a READ Continue macro instruction in BTAM. The user of READ requests in the any-mode must therefore explicitly request solicitation. Figure 5-25 illustrates both explicit and implicit soliciting of data.

Specific-mode and any-mode is also used when data is solicited. In the specific-mode, data is solicited only from a single terminal. In the any-mode, data is solicited from all connected terminals.

An application program might use these forms of solicitation in the following manner:

1. The application program initially solicits data from all of the terminals to which it has become connected.
2. The application program issues a READ in the any-mode which is completed when one of the terminals responds to the solicitation.
3. The application program communicates with the terminal using WRITE and READ macro instruction issued in the specific-mode. The READ macro instructions cause implicit solicitation to occur.
4. When the transaction is completed, the application program issues a new SOLICIT macro instruction specifically directed at the terminal, so that a new READ issued in the any-mode will be satisfied when the next transaction begins.

This technique of handling transactions is essentially the same as that shown previously in Figure 5-23, except that solicitation has been added. (Solicitation is not shown in Figure 5-23 because this figure is an example of communication with a logical unit, for which the concept of solicitation does not apply.)

When connection is established with a terminal in the basic-mode, the application program indicates the extent of data that each solicit request (implicit or explicit) is to obtain from that terminal. It is the application program's responsibility to determine when a new solicit request should be issued.

The application program can designate that for each solicit request, VTAM is to:

- Solicit only a *block* of data from the terminal. For start-stop terminals, a block ends with an EOB character; for BSC terminals, a block ends with an ETB or ETX character.
- Solicit a *message* from the terminal. Messages do not apply to start-stop terminals; for BSC terminals, a message ends with an ETX character. Messages consist of one or more blocks. Note that a message in basic-mode is not the same as a message in record-mode.
- Solicit a *transmission* from the terminal. For both start-stop and BSC terminals, a transmission ends with an EOT character. Transmissions are comprised of one or more basic-mode messages (for start-stop terminals, one or more blocks).
- Solicit the terminal *continuously* until the application program cancels the solicitation.

**Soliciting Blocks:** When data is solicited a block at a time and an error occurs during transmission, only a limited amount of data need be recovered by the terminal. However, since the application program must frequently reissue a solicit request (to acknowledge the previous block and obtain a new one), data throughput over the communications line is reduced. Block solicitation is appropriate when an unusually high number of line errors is expected and when the length of retransmitted data must be kept to a minimum, even if at the expense of slower response times and poorer line utilization. The installation must authorize, in the application's APPL definition statement, the solicitation of blocks.

**Soliciting Messages and Transmissions:** The lengths of basic-mode messages and transmissions are not as closely dependent on the type of terminal as are block lengths. Message and transmission lengths are usually established by the terminal's operator and the nature of the application. The lengths of messages and transmissions from a remote job-entry station, for example, are determined by the number of cards in each job deck, and the number of job decks available for transmission at one time, as shown in Figure 5-26.

Since messages and transmissions tend to be much longer than blocks, message and transmission solicitation means more data has to be recovered when an I/O error is detected. On the other hand, data transmission is much more efficient, because the acknowledgements and resolicitations needed to obtain the blocks are performed by the communications controller, not the application program.

Message and transmission solicitation is appropriate for applications that require short response times but can tolerate lengthy transmissions when required.

The selection of message solicitation over transmission solicitation (applicable only to BSC terminals) depends on how undesirable delays between messages would be or on whether communication is with conversational BSC terminals. With transmissions, delays between messages are minimized, although more data must be recovered if errors occur.

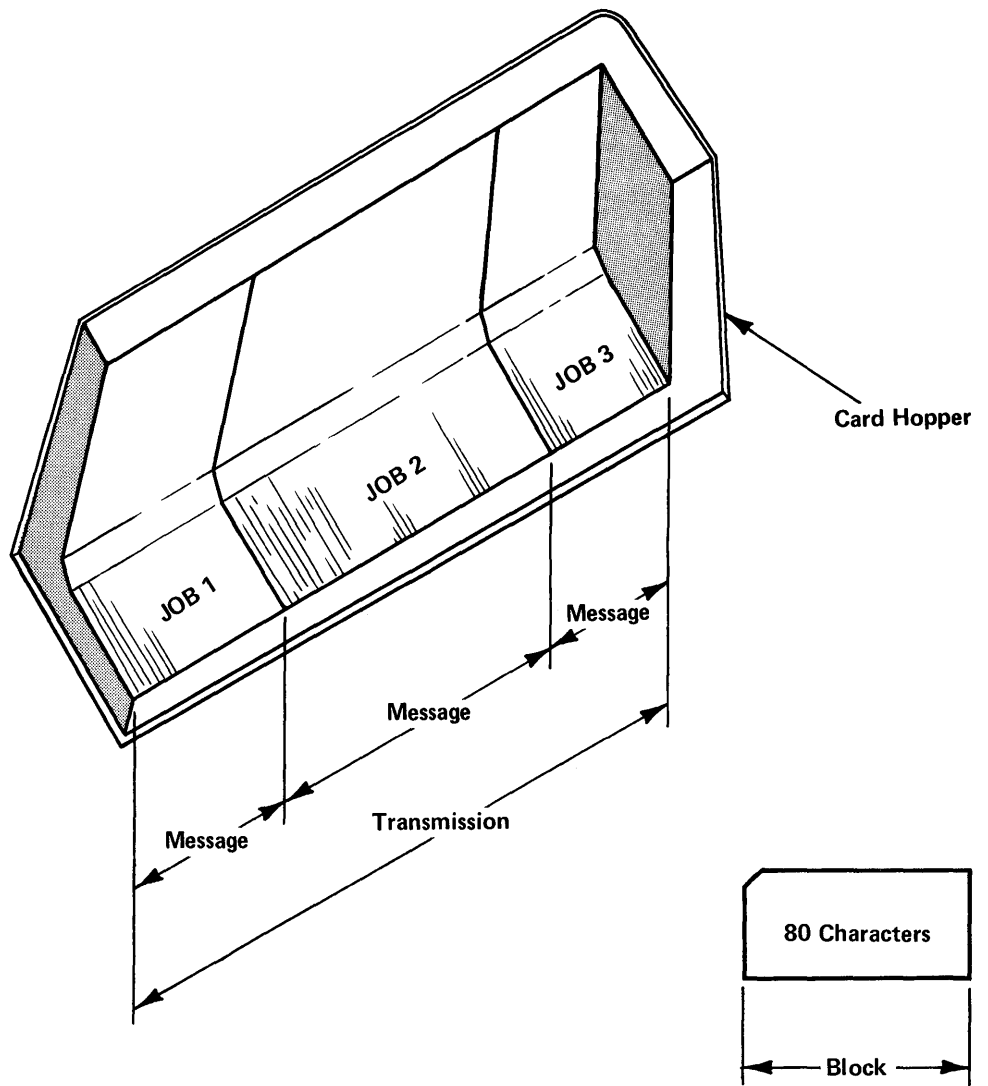


Figure 5-26. Basic-Mode Messages and Transmission from an RJE Station Vary in Length

**Continuous Solicitation:** The advantages and disadvantages of continuous solicitation are the opposite of those of block solicitation. By soliciting continuously, the application program can obtain data with the minimum of programming. However, the application program must determine when solicitation should cease, and must explicitly tell VTAM when to do so. If the solicitation must be interrupted frequently (for example, to send output to the same terminal), the efficiency is lost.

Continuous solicitation is appropriate for batch input applications, where transmissions are relatively frequent and delays between blocks, messages, and transmissions must be minimized.

## The VTAM Language

This section defines the functions of each VTAM macro instruction. It also explains how macro instructions and control blocks are related.



## ***Introduction to the VTAM Language***

The application program facilities described above are obtained by using VTAM macro instructions and by specifying desired values as parameters in these macro instructions.

VTAM provides assembler language macro instructions to request VTAM to:

- Connect or disconnect the application program to or from the VTAM network and then to specific logical units.
- Transfer messages and responses between the application program and a logical unit.
- Create control blocks to hold information that is passed between the application program and VTAM and to modify and interrogate these control blocks.
- Provide support for connection and communication activities.

The operands specified in these macro instructions are in keyword rather than positional format. Some of the operands must be specified; most operands are optional.

Most of the VTAM macro instructions rely upon parameters in the RPL. In addition to being able to modify these parameters via the MODCB macro instruction or assembler language instructions, the application program can change parameters as part of each individual connection, communication, or support request to VTAM.

In general, the VTAM language complements the VSAM language. Both VTAM and VSAM use ACB, EXLST, and RPL control blocks (although the formats of these control blocks differ in the two access methods). Both VTAM and VSAM have macro instructions (GENCB, MODCB, TESTCB, and SHOWCB) that are used to manipulate these control blocks.

Another common feature is the ability to code and specify the scheduling of exit-routines; these exit-routines are scheduled and executed independently of the other parts of the application program.

## ***The VTAM Macro Instructions***

The following provides a description of the macro instructions that were summarized earlier in this chapter. The macro instructions are grouped here, as they were earlier, into the following categories:

- Connection macro instructions.
- Communication macro instructions.
- Control-Block macro instructions.
- Support macro instructions.

## **The Connection Macro Instructions**

These macro instructions tell VTAM that a particular application program is in operation and, subsequently, request VTAM to connect the application to one or more logical units prior to transferring data. Macro instructions also request VTAM to disconnect the program from one or more logical units and to disconnect the program from the VTAM system.

- *OPEN*. Identifies an application program to VTAM. Once the program is opened, VTAM can schedule any exit-routines associated with the program. (The scheduling of a LOGON exit-routine also requires the issuance of a SETLOGON macro instruction.)
- *CLOSE*. Indicates to VTAM that an application program is terminating its connection with VTAM.
- *OPNDST*. Requests that VTAM connect the application program to a designated logical unit or list of logical units. Connection is required before communication macro instructions can be used to request data transfer with the logical unit.

- *CLSDST*. Requests VTAM to terminate the connection between the application program and a designated logical unit.

### The Communication Macro Instructions

These macro instructions request the transfer of messages and responses between the application program and a logical unit. Here is a brief description of the communication macro instructions:

- *RECEIVE*. Requests VTAM to transfer an available message or response from a specific logical unit or any one of a group of logical units to the application program's data area (if the input is data) or to one or more fields of the RPL (if the input is indicators or response information).
- *SEND*. Requests VTAM to transfer a message or a response to a specific logical unit. Data is transferred from an area in the application program; indicators or response information are specified symbolically in the SEND macro instruction in the RPL, and no output area is required.
- *SESSIONC*. Requests VTAM to send to a logical unit a SESSIONC indicator which either (1) starts or stops the possibility of exchanging messages with the SEND and RECEIVE macro instructions, or (2) assists in synchronizing the message sequence numbers being kept by a logical unit.
- *RESETSR*. Requests VTAM to change the continue-any/continue-specific mode of the logical unit. It also cancels any outstanding RECEIVE macro instructions that request input from the specific logical unit.

### The Control-Block Macro Instructions

The macro instructions described here are used to build and manipulate control blocks required by VTAM application programs. The first part describes the control blocks and lists the macro instructions used to assemble each when the application program is assembled. The second part describes the macro instructions that generate and manipulate the control blocks during program execution.

**Declarative Macro Instructions:** VTAM provides macro instructions to create these control blocks in the application program:

- Access method control block (ACB).
- Exit list (EXLST).
- Node initialization block (NIB).
- Request parameter list (RPL).

Here is what these control blocks are used for:

- *ACB* (Access Method Control Block). Contains information the application program provides VTAM about the application program in its entirety. Primarily, it names the application program and the list of exit-routines associated with the application. The ACB contains information about the *application program*.
- *EXLST* (Exit List). Contains the addresses of special exit-routines that VTAM is to schedule when certain conditions occur (for example, when a logon request is received from a logical unit). The EXLST contains the names of *exit-routines*.
- *NIB* (Node Initialization Block). Contains information the application program provides VTAM about general communication characteristics that are to exist between the application program and a particular logical unit. This information is provided to VTAM as part of a connection request; it remains in effect for the duration of a connection. The NIB contains information about a *logical unit*.
- *RPL* (Request Parameter List). Contains information (parameters) that an application program provides VTAM when requesting connection to a logical unit, input/output, and any action except the opening and closing of an ACB or the manipulation of a

control block. On completion of the requested action, the RPL contains information that VTAM passes to the application program. The RPL contains information about a *request*.

The ACB, EXLST, NIB, and RPL control blocks can be assembled into the application program by using the macro instruction associated with each control block, or they can be created and initialized during program execution, using the GENCB macro instruction. The type of control block that GENCB is to generate is specified in one of the operands in the macro instruction. The GENCB macro instruction is discussed below.

**The Manipulative Macro Instructions:** Like VSAM, VTAM provides a group of macro instructions that build control blocks or manipulate control-block fields. These macro instructions provide a more convenient release-independent way to do this than by using assembler language instructions. They refer to fields symbolically rather than by specific control-block location. The manipulative macro instructions are:

- *GENCB*. Builds an ACB, EXLST, NIB, or RPL during program execution and can initialize designated fields with specified values.
- *SHOWCB*. Obtains the value or values from one or more fields of a control block and places them in an area in the application program where they can be examined. In addition to fields that are set by the application program's use of macro instruction keyword operands, a number of control block fields can be shown that are set by VTAM but that cannot be directly modified by the application program.
- *TESTCB*. Tests the contents of a field against a value and accordingly sets the condition code in the program status word (PSW).
- *MODCB*. Changes the contents of one or more fields by inserting specified values in the fields.

There are several different forms of the manipulative macro instructions. In addition to the standard form, there is a list form, a remote list form, a generate form, and an execute form. The nonstandard forms can be used for programs that must be reenterable or that are sharing with other programs the parameter lists that are assembled when the macro instructions are expanded.

Instead of using the manipulative macro instructions, a VTAM application can manipulate values in these control blocks by using DSECT and other assembler language macro instructions. While less convenient to code, fewer instructions will be executed.

## Support Macro Instructions

VTAM provides these additional macro instructions to support connection and communication activities:

- *CHECK*. Checks and, if necessary, awaits completion of a previously requested RPL-based operation, marks as inactive the RPL associated with the request (thus freeing it for further use), and, if a logical or other error is detected and a LERAD or SYNAD exit-routine exists, causes the appropriate routine to be called.
- *EXECRPL*. Requests VTAM to perform an operation that is currently specified in a designated RPL. EXECRPL can be used instead of using other RPL-based macro instructions such as OPNDST, SEND, and RECEIVE, or, after an unsuccessful operation, when reissuing a request. Prior to issuing an EXECRPL, the operation to be performed may already be specified in the REQ field of the RPL or the application program may specify it. Any other modifiable field of the RPL can be set up prior to issuing the EXECRPL macro; in addition, any valid RPL-based parameter may be specified in an EXECRPL macro instruction operation. While less convenient to set up and code, the use of EXECRPL requires that fewer instructions be executed.

- **INQUIRE.** Obtains certain information that the application program may need and places it in a specified area of the program. The information that can be requested using INQUIRE includes: the logon message associated with a logon request, the number of logical units currently connected, or queued for logon to, the application, the physical address of a 3270 screen (for use in a copy operation), and whether another application program is active or inactive and whether it is accepting logon requests.
- **INTRPRET.** Provides a means of accessing an installation-defined interpret table. For example, INTRPRET can be used to obtain the real symbolic name of an application program when the program is identified with an alias in a logon message. INTRPRET can be used by special programs written to receive logon messages and reconnect logical units to the appropriate application program.
- **SETLOGON.** Tells VTAM to activate the application program's LOGON exit-routine and to begin queuing any logon requests. The user can also temporarily halt the queuing of logon requests until more logical units can be handled or can permanently halt the queuing of logon requests in preparation for a close-of-day operation.
- **SIMLOGON.** Allows the application program to acquire a logical unit by initiating the logon request on behalf of one or more logical units to which the application wants to be connected.

### ***Relating VTAM Control Blocks and Executable Macro Instructions***

The control blocks created by VTAM macro instructions are used to pass information between the application program and VTAM. One control block, the EXLST, is used simply to pass the names of exit-routines in the application program to VTAM. The other control blocks, the ACB, RPL, and the NIB, are used both to pass information to VTAM and to contain information that VTAM returns to the program.

The most frequently used control block is the RPL. An RPL is required each time the program makes a request for a connection, for an I/O operation, or for any service except opening and closing an ACB. The other control blocks, the ACB, EXLST, and NIB, may be used as little as once during the execution of the program.

Each RPL-based macro instruction refers to an RPL that contains information about the operation to be performed, such as the address of the input or output area, the length of the area, and the logical unit to be communicated with. This information is placed in the RPL initially when the control block is created or subsequently, either by using a MODCB macro instruction or by providing the information in the request macro instruction itself. For example:

```
RECEIVE RPL=RPL1,AREA=INFO,AREALEN=200
```

would change the value in RPL1's AREA and AREALEN fields to INFO and 200 respectively before initiating the input operation. These values would remain in these fields for all subsequent requests that used RPL1 until they were changed again.

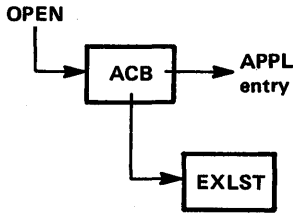
The relationship of the VTAM control blocks to each other and to the macro instructions that refer to them can be described in the context in which they are used:

- Opening the application program—that is, identifying itself to VTAM as operational.
- Connecting to logical units with which it will communicate.
- Communicating with connected logical units.

These activities as they relate to the VTAM language are discussed below.

### **Opening the Application Program**

The OPEN macro instruction indicates to VTAM that a program is now an operational part of the VTAM network. The OPEN specifies an ACB; the ACB in turn points to a

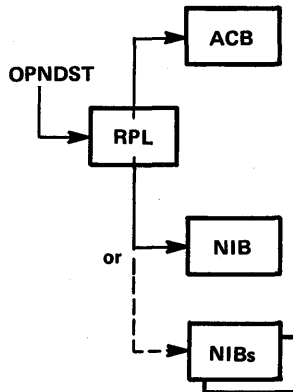


location in the program that contains the name of the application program as defined in an APPL statement during VTAM definition. The ACB may also point to an EXLST control block containing the names of exit-routines that are to be associated with the application program. (An EXLST can also be pointed to when a logical unit is connected; see "Connecting Logical Units", below.) When the open process is complete, any exit-routines that have been specified are eligible for scheduling by VTAM.

More than one ACB can be opened at a time by a single OPEN macro instruction. This means that a program that performs related functions (for example, communicating with both logical units and other terminals) may be defined so that it is viewed by VTAM as more than one application program. Probably most VTAM users will find it satisfactory to open only one ACB per program.

The CLOSE macro instruction notifies VTAM that an application is detaching itself from the VTAM system. As a result of issuing a CLOSE macro, any logical units still connected to the program are disconnected.

### Connecting Logical Units



Before communicating with a logical unit, an application program must have itself connected to the logical unit. Either the logical unit or the application program can initiate connection; in either case, it is the application program that formally requests connection, by using an OPNDST macro instruction. The OPNDST macro instruction specifies an RPL that is associated with the request. The RPL contains the address of a NIB. The NIB contains information that applies to subsequent communication with the logical unit; this information can include a unique storage address to be associated with the logical unit. If a number of logical units are to be connected, a single OPNDST can be used, with the RPL pointing to a list of NIBs instead of to a single NIB.

Optionally, a NIB can point to a list of exit-routine names in an EXLST control block. For the logical unit being connected, these exit routines are used in preference to any equivalent exit-routines identified for the entire application program when the ACB was opened.

When a logical unit is connected as the result of an OPNDST macro instruction, VTAM returns information about the logical unit in the RPL and the NIB. In both the RPL (if only one logical unit is connected) and the NIB, VTAM places a communications identifier (CID) that it has assigned to the logical unit. The CID is a network address; it is shorter and requires less space and search time for VTAM than the symbolic name of the logical unit specified during VTAM definition. On all subsequent I/O requests for the logical unit, the application program must be sure that this CID is located in the RPL. In addition to the CID, VTAM also places the logical unit name, characteristics, and other information in the NIB; if desired, the application program can use this information to determine how to communicate with the logical unit.

Once a NIB has been used to connect one logical unit, it can be reinitialized and reused for connection with other logical units.

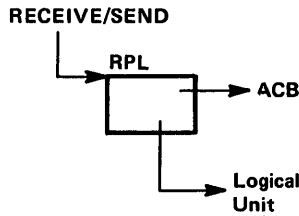
### Disconnecting Logical Units

Once a series of communications between the application program and a logical unit is complete, the program disconnects the logical unit using the CLSDST macro instruction.

If the program is terminating and all logical units are to be disconnected at the same time, the CLOSE macro instruction, used to close the ACB, also disconnects all connected logical units and the CLSDST macro need not be used.

### Communicating with Logical Units

Having opened the application program's ACB and having connected one or more logical units to the program, the program can communicate with each connected logical unit by



issuing SEND and RECEIVE macro instructions. VTAM obtains the name of the application program that made the request and the identity of the logical unit (if a specific logical unit is being addressed) from the RPL. The communication macro instruction specifies an RPL; the RPL contains the address of an ACB and the identity of the logical unit.

Only data is written from or read into an application program data area. Indicators and response information are sent as a result of being specified symbolically in a SEND macro instruction or in an RPL. Indicators and response information that are received are not read into a data area but are detected by analyzing fields in the RPL associated with a RECEIVE macro instruction or in an RPL associated with the scheduling of an exit-routine specified to handle the receipt of indicators or response information.

VTAM provides two ways to write to a logical unit. A SEND macro instruction can be issued that specifies POST=RESP, in which case VTAM handles any response signals that are returned and notifies the application program when the requested operation is complete; this is called responded output. (If necessary, the application program can obtain response information from the RPL.) A SEND macro instruction can also be issued that specifies POST=SCHED, in which case VTAM considers the SEND complete as soon as the output operation has been scheduled; this is called scheduled output. The application program thus has the responsibility for determining completion either by having a RECEIVE macro instruction that specifies response information is to be received or by having a RESP exit-routine that VTAM schedules each time response information arrives.

### *Handling Control Blocks, I/O Areas, and Work Areas*

The application program can handle control blocks in a number of ways. It can:

- Define RPLs, NIBs, or EXLSTs in the application program during assembly or generate them, using the GENCB macro instruction, during program execution.
- Assign one RPL or NIB to a specific logical unit during assembly, or either assemble or generate RPLs and NIBs that are to be available for any logical unit as the need arises.
- Retain the RPL used in connecting the logical unit for all further communication with the logical unit.
- Use one RPL for all connection requests and use another RPL or group of RPLs for all communication requests.
- Define the RPLs, NIBs, and any other required control blocks or work areas to be associated with logical units as a pool so that a limited amount of control block storage is not exceeded.

In application programs that must handle many logical units concurrently, it may be useful to have a control block work area other than the RPL or NIB associated with a particular logical unit or logical unit conversation. The queuing of input or output by the application program between its processing routines and logical units may also require a logical unit-related control block work area. VTAM provides a field in the RPL, the user field, that may be used to associate storage with a particular logical unit. The application program initially associates the storage with the logical unit when the logical unit is connected by specifying an address in the USERFLD of the NIB; thereafter, when input is received from the logical unit VTAM provides the address associated with it in the RPL's user field.

### *Coordinating I/O Activity*

The VTAM user has several ways to coordinate the transfer of data to and from logical units with the preparation and processing of data by the program. The simplest way is to make the program wait each time an I/O operation is requested until that operation is completed. This is called synchronous request handling and is specified by coding

OPTCD=SYN in the I/O request macro instruction or in the associated RPL. This method is satisfactory for the transmission of data between an application program and one or a few logical units, or for reading a message from a master logical unit where all further program action depends on analysis of that message.

Most programs, however, require asynchronous request handling for effective processing. Asynchronous request handling is specified by coding OPTCD=ASY in the I/O request or in the associated RPL. In this type of operation, the program continues to execute after an I/O request is accepted by VTAM. On completion of the requested operation, VTAM, as specified by the program, either schedules an exit-routine associated with the request (specified by coding the EXIT operand in the macro instruction or associated RPL) or posts an ECB (specified by coding the ECB operand). If an RPL exit-routine is specified, it is scheduled automatically for the application program. An ECB, if specified, must either be checked periodically to see if it is posted, or the application program can be put into a wait state, using either a system WAIT macro instruction or VTAM's CHECK macro instruction. A WAIT macro instruction can be for one or any one of list of ECBs; a CHECK can wait only for the completion of the operation associated with one RPL. (Whether an RPL exit-routine or ECB posting is used, a CHECK must be used to make the RPL available for reuse.)

### ***Communicating with Start-Stop and BSC Terminals***

The control blocks and macro instructions described previously (except for the RECEIVE, SEND, SESSIONC, and RESETSR macro instructions) apply as well for communication with terminals on start-stop or BSC lines. This section describes the additional macro instructions and language facilities that apply when communicating with these terminals. These macro instructions are used to communicate with terminals in basic-mode. Readers who do not plan to use basic-mode in a VTAM application program may wish to skip this section and go to the next section, "Sample Programs," in this chapter.

### **Distinguishing Between Logical Units and Start-Stop/BSC Terminals**

Prior to connecting a terminal, the INQUIRE macro instruction can be used to determine whether the terminal is a logical unit or a start-stop/BSC device. The symbolic name of the terminal is furnished by VTAM when a LOGON exit-routine is entered. This name must be placed in the NIB pointed to by the RPL specified in the INQUIRE. VTAM places the device characteristics in an area specified by the application program. By testing the value in this area, the application program can determine the appropriate macro instructions and related logic to use for communication with the start-stop/BSC terminal or logical unit.

### **The LDO Control Block**

In addition to the control blocks described previously in "The Control Block Macro Instructions," VTAM application programs that communicate with certain start-stop and BSC terminals can define a logical device order (LDO) control block. This control block defines a particular kind of I/O operation that is not ordinarily performed, such as writing a positive response with leading graphics to a System/3 or System/370 CPU. The operation is requested by issuing a DO macro instruction that specifies the LDO.

### **The CHANGE Macro Instruction**

In addition to the macro instructions described previously in "Support Macro Instructions," another macro instruction, CHANGE, can be used only with terminals in basic-mode. The CHANGE macro instruction requests that the information furnished to VTAM as part of a connection request be changed. The macro instruction assumes that the NIB, used to pass information when requesting connection, has previously been modified with a MODCB macro instruction.

## The Basic-Mode Communication Macro Instructions

The communication macro instructions for start-stop and BSC terminals are called *basic-mode* macro instructions. (The communication macro instructions for logical units are *record-mode* macro instructions.) Here is a brief description of the basic-mode communication macro instructions:

- **SOLICIT.** Requests VTAM to solicit input from a specific start-stop or BSC terminal or from a group of terminals. Input is read into a VTAM buffer, not into the application program; a READ macro instruction is used to read the input into the application program's data area. The solicitation action caused by a SOLICIT to a group of terminals continues until input has been received from every terminal in the group. Once input has been received from a terminal, the terminal must be resolicited unless, at connection, continuous solicitation of the terminal was specified.
- **READ.** Requests VTAM to transfer data from a specific start-stop or BSC terminal or from any one of a group of terminals into an area in the application program. A request to read any one of a group requires a SOLICIT prior to the READ; a request to read a specific terminal causes solicitation of input to take place if a SOLICIT was not previously issued for that terminal. If data is already in a VTAM buffer as the result of a previous solicit or read operation, the transfer of data takes place immediately.
- **WRITE.** Requests VTAM to transfer data from an application program to a specific start-stop or BSC terminal. The application can also request that VTAM send certain control information to a terminal or write conversationally (write and then read from a terminal).
- **DO.** Requests VTAM to perform the special I/O action associated with a specific LDO control block in the application program.
- **RESET.** Requests VTAM to cancel all outstanding I/O requests to a start-stop or BSC terminal and, if necessary, to reset any error lock that may have been set for the terminal to prevent output.

## Sample Programs

This section shows and discusses the general logic of two sample application programs. Following the discussion of the sample programs is a discussion on VTAM alternatives to programming techniques used in the samples.

### *Introduction to the Sample Programs*

The sample programs in this section show how VTAM macro instructions are used in VTAM application programs. Many, but not all, VTAM facilities and language features are illustrated in these sample programs. In the text that explains the programs, some alternative facilities or techniques are discussed, but they are not shown in the figures.

Sample Program 1 shows how simple a VTAM program can be; it is not designed to portray a typical program.

Sample Program 2 illustrates additional VTAM programming concepts and techniques and contains more detail than Sample Program 1. Sample Program 2 communicates with logical units in a 3601 Finance Communication Controller and with 3270 Information Display System terminals. It communicates with both types of terminals using the record-mode macro instructions (SEND and RECEIVE).

The logic of each program is displayed in flowcharts that show the major decision points but do not show all program instructions. However, the key macro instructions and operands are shown, and they are discussed in notes that are keyed to each figure. Programming examples are provided throughout this section. Language options are explained in the text as they are introduced.



To follow the sample programs, the reader must be familiar with programming techniques for the acquiring and handling of control block areas, the posting of ECBs, and the scheduling of processing routines by a telecommunication program. Following Sample Program 2, in "Choosing Programming Alternatives Discussed in the Sample Programs," is a discussion of alternatives to some of the programming techniques used in the sample programs.

## Sample Programs

Figure 5-27 shows a VTAM application program that would handle a request for logon (connection) for a logical unit, connect it, read input from any connected logical unit, process the input, prepare a reply for output, and then write the output to the logical unit.

Sample Program 1 demonstrates use of:

- A LOGON exit-routine and the acceptance of a logon request.
- A request to receive input from any connected logical unit.
- Synchronous I/O requests.
- Continue-any and continue-specific modes.
- A SEND macro instruction to send a response rather than data.
- A request to *schedule* output.
- A RESP exit-routine to handle a response received from the logical unit each time it receives data.
- A TPEND exit-routine.

For simplicity, error recovery routines and other special routines are omitted; these are discussed in Sample Program 2.

Sample Program 1 might be feasible for a VTAM application program for which each transaction between the application and the logical unit consisted of a short inquiry and a short reply. Because the application program waits for the processing for one logical unit to complete before reissuing a request for input from any connected logical unit, the application might not adequately serve a large number of logical units communicating with the application program at the same time.

### The Logic of Sample Program 1

These notes are keyed to Figure 5-27.

- 1 The ACB, defined in the program with the ACB macro instruction, is opened with the OPEN macro instruction. For example, this might be coded:

```
OPEN      ACB1
```

ACB1 contains:

```
ACB1      ACB          AM=VTAM,APPL=APPL1,EXLST=EXLST1,
                          MACRF=LOGON
```

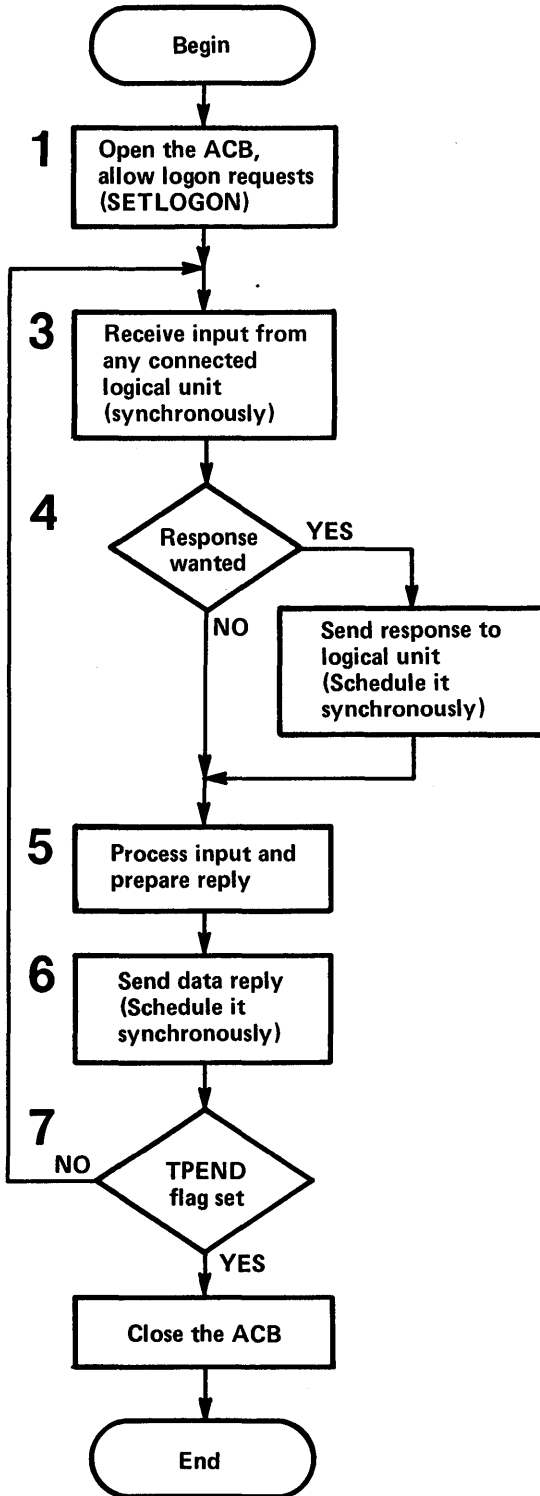
APPL1 contains:

```
APPL1     DC          X'05'
          DC          C'PROG1'
```

PROG1 is the name of the APPL statement used to define the application during VTAM definition.

EXLST1 contains the names of the exit-routines shown in Figure 5-27. MACRF=LOGON indicates logon requests are accepted.

### Main Program



### Exit Routines

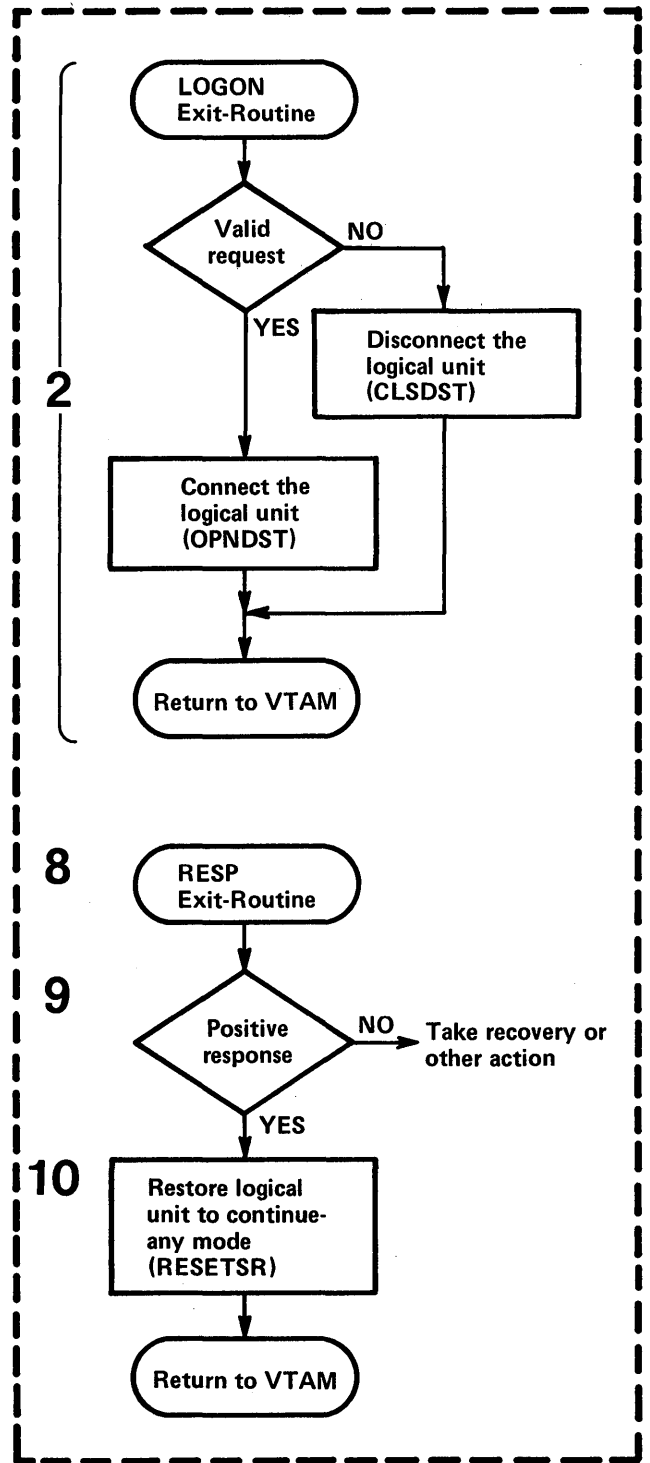


Figure 5-27. The Logic of Sample Program 1

After the OPEN macro instruction, a SETLOGON macro instruction is used to tell VTAM to begin processing logon requests for the application. The macro instruction might be coded:

```
SETLOGON RPL=RPL1,OPTCD=START
```

- 2 The LOGON exit-routine, whose address is specified in the LOGON operand of the EXLST macro instruction (whose address is in turn specified in the ACB macro instruction), is scheduled when VTAM receives a request for connection from or on behalf of a logical unit. Providing no other exit-routine is currently being executed, the LOGON exit-routine is given control. When the LOGON exit-routine is completed, either the next-scheduled exit-routine receives control or the main program regains control at its next sequential instruction.

An INQUIRE macro instruction could be used to obtain the logon message when there is one. By examining the logon message, the exit-routine determines whether the logical unit should be connected to the program. If so, an OPNDST macro instruction like this might be issued:

```
OPNDST RPL=RPL1,OPTCD=(ACCEPT,SPEC),NIB=NIB1
```

An RPL and a NIB (RPL1 and NIB1, respectively, in this example) are required for the OPNDST request; the same RPL and NIB can be reused by the OPNDST each time the LOGON exit-routine is entered. If necessary, the type of terminal and other communication characteristics can also be obtained using the INQUIRE macro, and the NIB properly initialized with this information prior to issuing the OPNDST macro. (The MODCB macro instruction can be used to initialize the NIB.)

OPTCD=(ACCEPT,SPEC) specify options of the OPNDST requesting that a logon request from a *specific* terminal (identified in the NIB) be *accepted*.

If the logical unit is not to be allowed to use the program, a CLSDST macro instruction must be issued to notify VTAM that the logical unit's logon request is being rejected. Although not shown in the flowchart, it may be desirable to connect the logical unit (using OPNDST), write an appropriate message to it, and then disconnect.

- 3 In this example, the first request to receive input from any connected logical unit is issued in the main program. In this case synchronous request handling is specified; that is, the application program indicates that it will wait until input is received from one of the logical units. In making the request, the application identifies an RPL and an input area. The macro instruction might be coded:

```
RECANY RECEIVE RPL=RPL1,AREA=AREA1,AREALEN=100,  
RTYPE=DFSYN,OPTCD=(SYN,ANY,CS)
```

or it could be coded:

```
RECANY RECEIVE RPL=RPL1
```

where RPL1 was coded:

```
RPL1 RPL AM=VTAM,ACB=ACB1,AREA=AREA1,  
AREALEN=100,RTYPE=DFSYN,  
OPTCD=(SYN,ANY,CS)
```

The input request could have been coded with some operands appearing in the RPL and others in the macro instruction. If an operand that appears in the RPL is also coded in the macro instruction, the value in the macro instruction replaces the value in

the RPL and is in effect not only for the current operation but for subsequent operations that use the RPL.

AM=VTAM indicates the RPL is to be used for a VTAM request. ACB=ACB1 specifies that the application program issuing the request is using ACB1.

AREA=AREA1 specifies that any input data resulting from the request should be moved to the application's input area AREA1 which has a length of 100 bytes (AREALEN=100).

RTYPE=DFSYN is specified so that only the receipt of data or of a synchronous-flow indicator completes the request. (The receipt of a response causes the RESP exit-routine to be entered; see 8 below.)

CS is specified so that the logical unit whose input is read by the RECEIVE is put in continue-specific mode until its inquiry has been successfully answered. Thus, if the logical unit is still in continue-specific mode when the next RECEIVE with OPTCD=ANY is issued, input from that logical unit does not satisfy the any-input request. This allows input from other logical units to be processed; a single, busy logical unit does not monopolize the program. The logical unit is put back in continue-any mode when a response has been received acknowledging that the reply to the inquiry arrived successfully; in this sample program, this is done in the RESP exit routine.

Assume that a logical unit just connected sends in the first inquiry to the program. The synchronous RECEIVE completes. If register 15 contains 0, the operation was successful. If register 15 contains some other value than 0, an error or special condition has occurred. A LERAD or SYNAD exit-routine in the program may have been entered, and may have returned a code in register 15 or register 0 that will indicate further action for the program to take. Whether a LERAD or SYNAD were entered, information is available in various feedback fields of the RPL for analysis. One of the errors that can occur is that the message arrived as an exception; this information is available as one of the feedback return codes in the RPL.

Assuming the operation was successful, the identity of the logical unit whose input was received is provided in the ARG field of the RPL. Data is located in AREA1, and a SHOWCB or TESTCB can be used to determine its length (by examining the RECLLEN field of the RPL).

- 4 The logical unit that sent the data may have indicated that the program should issue a response to verify that the input has been received.

There may be some occasions when the logical unit wants a response, perhaps to verify that a data base update message has been received so that it can free its buffers. On other occasions such as an inquiry message, the logical unit may not want a response (or want a response only if an exception condition occurs); the answer to its inquiry will be forthcoming soon and will be implicit assurance that the inquiry arrived. The VTAM application program can examine the RESPOND field of the RPL to determine whether and under what conditions a response is required. If completion information following the RECEIVE indicates that the input was received normally and the RESPOND field indicates that a normal response is required, it is sent with a SEND macro instruction that might be coded:

```
SENDRESP SEND RPL=(2),STYPE=RESP,OPTCD=SYN
```

If completion information following the RECEIVE indicates that an exception message was received (in which case there will be no input to process), and the RESPOND field indicates that a response is requested, it is sent with a SEND macro instruction that might be coded:

```
SENDRESP SEND RPL=(2),STYPE=RESP,OPTCD=SYN,RESPOND=EX
```

If the logical unit wanted a response only in the event of an exception, the RESPOND field will already be set to EX and will not have to be reset in the SEND. Before issuing the SEND for the exception response, the program would set up sense information in the RPL, defining the exception.

The same RPL used for the RECEIVE request can be reused for the SEND. Since a response is being sent (STYPE=RESP), no data area or length is needed. For a response, POST=SCHED completion is assumed. The request is specified for synchronous handling (OPTCD=SYN). However, because it is only being scheduled, the operation takes a relatively short time. As soon as the operation has been scheduled and the SEND completes, the RPL can be reused.

- 5 The inquiry is analyzed and a reply is prepared by a processing routine. Disk I/O may be required. If so, the program waits until the reply is ready. (This sample program assumes that this processing time is acceptably short.)
- 6 The reply is then sent with a SEND that requests the transmission of data (STYPE=REQ). In this sample program, the same area used to receive input is used for output. The macro instruction might be coded:

```
SENDATA SEND RPL=(2),STYPE=REQ,RESPOND=(NEX,FME),  
OPTCD=SYN,POST=SCHED
```

Since the RPL currently contains the address of AREA1 in the AREA field, AREA=AREA1 need not be respecified in the macro. So that the application program can determine whether the logical unit receives the data successfully, a response from the logical unit is requested (RESPOND=(NEX,FME)). NEX indicates that either a normal or an exception response is to be returned (one or the other). Although not shown, EX indicates only an exception response is to be returned, and NFME indicates no response at all is to be returned. The macro instruction requests the scheduling of the operation; the request is completed synchronously (OPTCD=SYN) as soon as the sending of the output has been scheduled (POST=SCHED). Completion of the operation is determined as a result of the program's receiving the FME response.

With the reply under way, a branch is made back to the RECEIVE so input that may have been read into VTAM's buffers from another connected logical unit (that is not in continue-specific mode) can be read into the application program for processing.

- 7 If the VTAM network is being halted, the application program's TPEND exit-routine (not shown) is scheduled and entered by VTAM. This routine can indicate to the main portion of the application program that the application is to terminate. The TPEND exit-routine or later the main program can send final messages to connected logical units and do other close processing depending on whether the closedown is quick or a routine end-of-day (normal) closedown. The main program, discovering the closedown requirement, must issue a CLOSE macro instruction to disconnect the application program from the network; any logical units not yet disconnected are, as a result of the CLOSE, disconnected. Note that the CLOSE macro instruction must be issued in the main program and not in the TPEND exit-routine. The application program terminates by returning control to the operating system.

- 8 When a response to the data sent in 6 is received by VTAM, the RESP exit-routine is scheduled and entered. On entry, register 1 points to a parameter list that points to a read-only RPL in VTAM's storage, whose feedback fields can be examined to determine the kind of response received.
- 9 If an exception response has been received, the application program can determine from sense information in the RPL whether or not to retry the operation, disconnect the logical unit, or take some other action.
- 10 If a positive response is received, the logical unit can be returned to continue-any mode so that the next request for input from any logical unit includes this logical unit as one whose input can be read by the VTAM application program. This is done by issuing:

```
RESETSR      R=(2),OPTCD=CA,RTYPE=DFSYN
```

The RESP exit-routine must have its own RPL available. The address of the RPL is placed in register 2 in this example. The identity of the terminal to be placed back in continue-any mode must be put in the ARG field of the exit routine's RPL. The RESP exit-routine then returns control to VTAM.

### *Sample Program 2*

Sample Program 2 is a more typical example of a VTAM application program than Sample Program 1. Sample Program 1 should be read first to gain an understanding of some of the elementary concepts of using VTAM macro instructions.

Sample Program 2 communicates with work stations (VTAM logical units) in a 3600 Finance Communication System and 3270 Information Display System devices. Both kinds of terminals (logical units and 3270s) are physically connected to VTAM on nonswitched lines through a 3704 or 3705 Communications Controller with a network control program. Additionally, local 3270s can be attached directly to VTAM through a channel. Although the VTAM application program uses the record-mode to communicate with both types of terminals, the logical units are on lines that use SDLC, and the 3270 devices are on lines that use BSC or are locally attached via data channels.

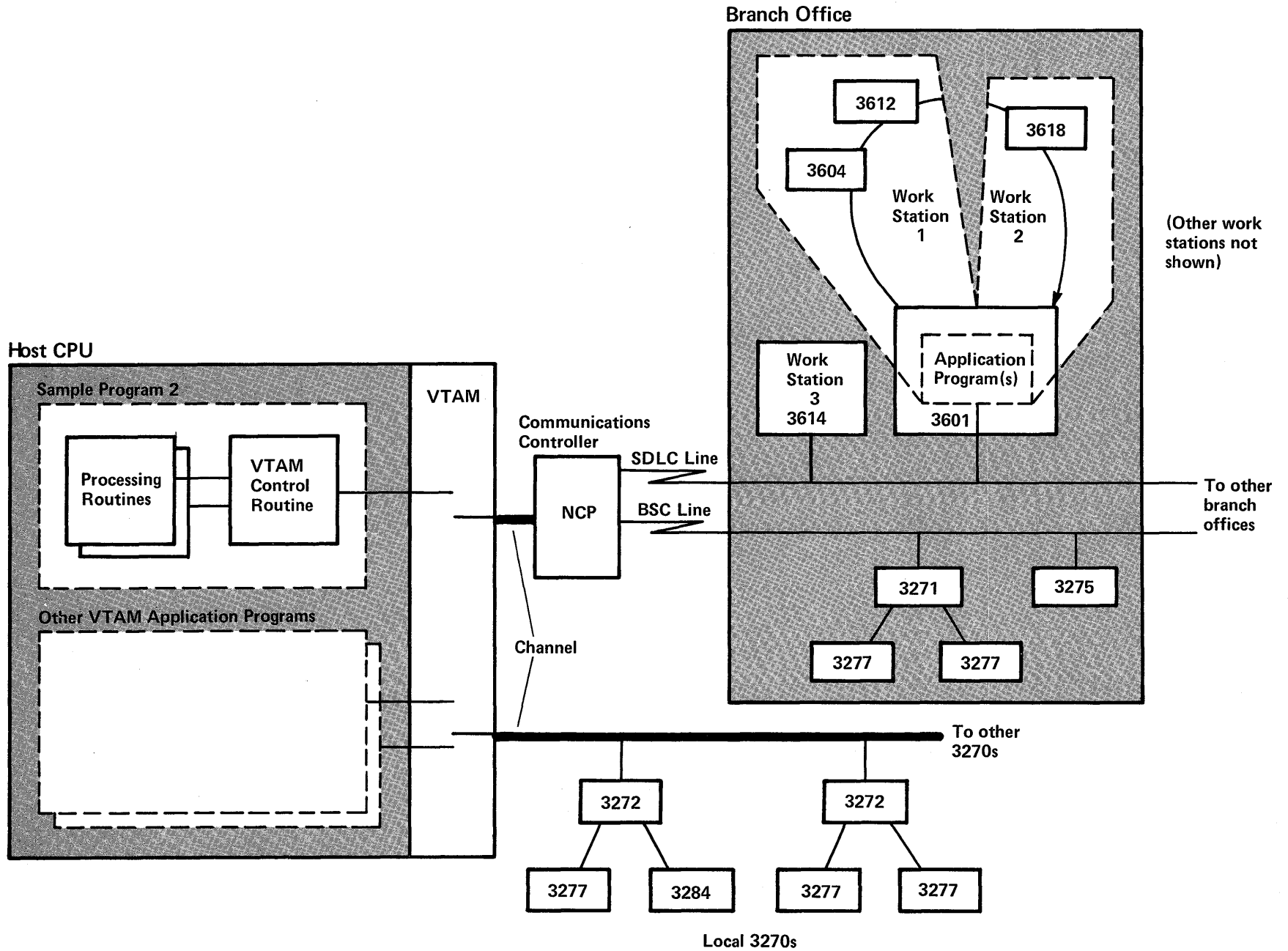
Figure 5-28 shows a possible configuration of terminals with which Sample Program 2 might communicate.

Note in Figure 5-28 that the application program in the 3601 controller can be written to perform certain functions that would otherwise have to be performed by the VTAM application program. For example, the 3601 application program could be written to screen inquiries for correct format prior to forwarding them to the VTAM application program for processing, or it could be written to collect inquiries from several devices that form a work station and send them as one transmission to the VTAM application program.

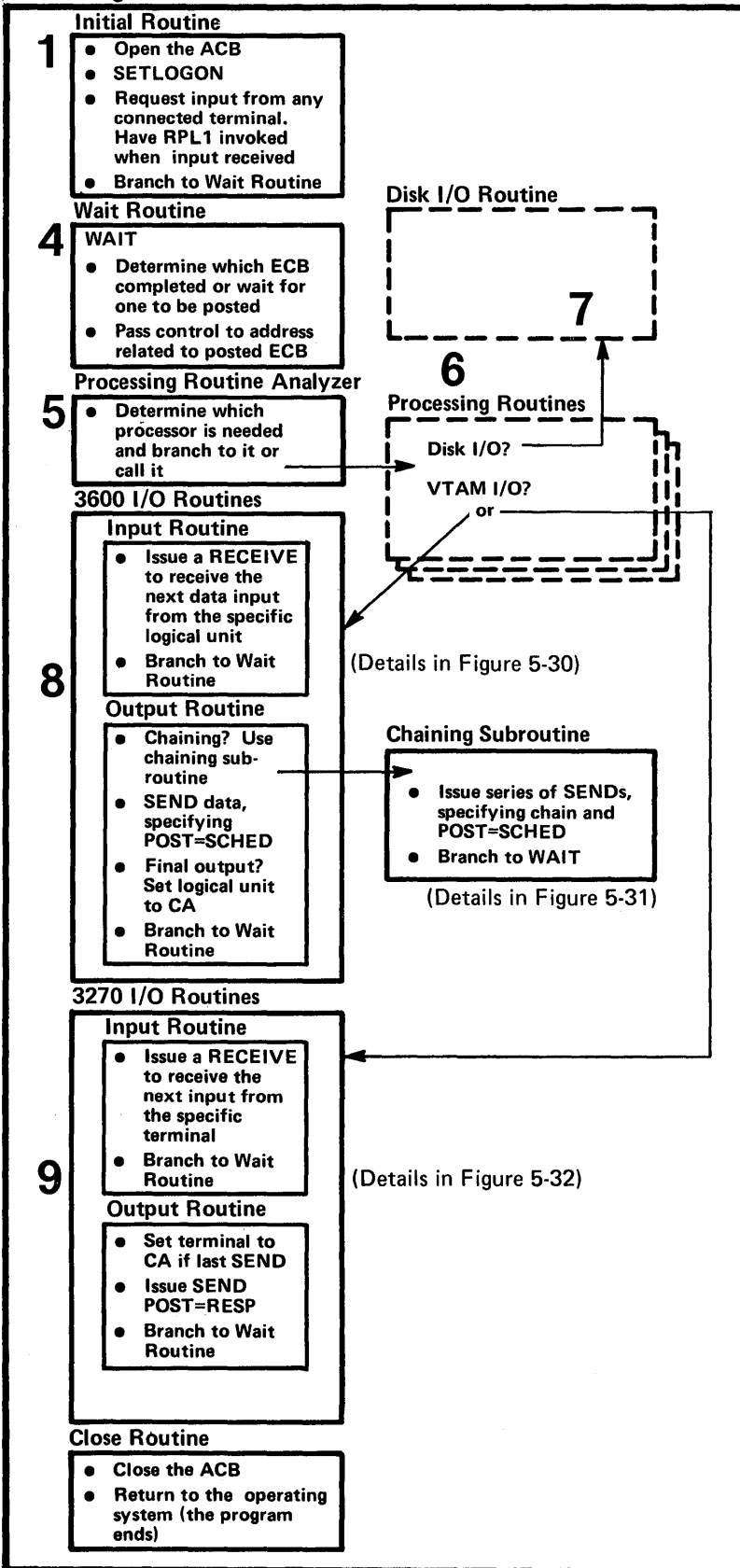
The logic of Sample Program 2 is described at a high level in Figure 5-29 and accompanying notes. The logic of special routines is described in more detail in subsequent figures and accompanying notes.

Sample Program 2 uses the posting of ECBs, either by VTAM or within the application program, and a central wait routine that detects posted ECBs as a mechanism to handle a number of terminals alternately without having to suspend all program execution while waiting for I/O operations to be completed. After a request has been issued for an operation to be performed asynchronously to the main program, control is transferred to the wait routine, which detects (or, if necessary, waits for) a posted ECB. When the ECB is posted, the wait routine determines the event that completed and branches to a point

Figure 5-28. Hypothetical Network-Configuration for Sample Program 2



## Main Program



## Exit Routines

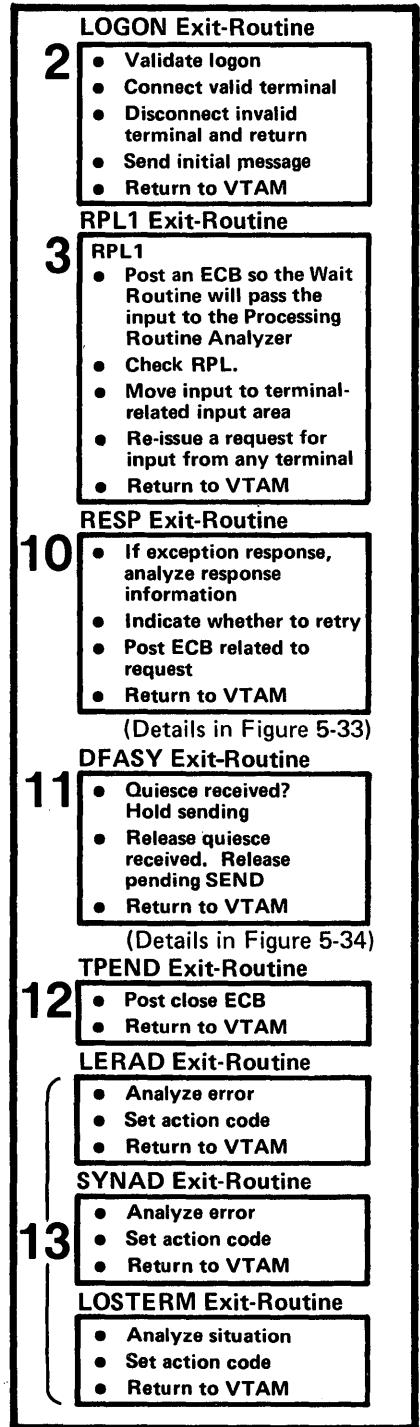


Figure 5-29. Organization and Flow of Sample Program 2



in the program to provide further processing for the event. An understanding of the details of this technique is assumed in this discussion.

Although not discussed in detail, it is likely that Sample Program 2 would use a separate work area (or control block) for each terminal that is actively using the program. This work area might include an input/output area. A separate RPL might be associated exclusively with each active terminal. The storage for the RPL and the work area might be obtained from a fixed pool or be obtained dynamically and the RPL initialized using the GENCB macro instruction. This work area and RPL area might be obtained and related to a terminal for the duration of its connection, for the duration of the program, for the duration of a transaction or conversation, or on some other basis. The ECB associated with a terminal could be located in the RPL or outside of it in some fixed relationship, perhaps just in front of it. In Sample Program 2, it is assumed that the storage for an ECB, RPL, and work area is obtained and initialized in the LOGON exit-routine and retained for the duration of the terminal's connection to the application program.

### The Organization and Flow of Sample Program 2

Figure 5-29 shows the principal routines in Sample Program 2; the notes below indicate how Sample Program 2 works. More detailed logic is shown and discussed in subsequent figures and notes.

Figure 5-29 shows the main portion of the application program (all parts of the application program other than the exit-routines) and the exit-routines as separate groups of routines. This is a logical rather than a physical separation; exit-routines are distinctive because they are entered only when an event occurs that requires handling by an exit-routine. VTAM suspends execution of the main program until the exit-routine completes its processing and returns to VTAM. Only one exit-routine can execute at a time; if an exit-routine event occurs while an exit-routine is executing, the second exit-routine is scheduled for entry only after the first exit-routine is completed. The LERAD and SYNAD exit-routines are exceptions to this general rule; they can be entered as the result of a synchronous I/O operation or the issuance of a CHECK macro in another exit-routine (in which case, they may be viewed as extensions of the exit-routine that caused them to be entered).

Except for the LERAD and SYNAD exit-routines, each exit-routine must establish its own addressability, execute, and then return to VTAM; VTAM's registers need not be saved or restored. A temporary branch to part of the main program could be made from an exit-routine—common code could be shared—but the exit-program routine is considered to be in progress until control is returned to VTAM. The LERAD and SYNAD exit-routines are furnished addressability as the result of loading registers (the user save-area address is passed in register 13).

Except for an RPL exit-routine, whose address is specified in the RPL or request macro instruction, the addresses of the exit-routines to be associated with the program are defined in an EXLST macro instruction. In addition, for three types of exit routine—DFASY, RESP and SCIP—different exit-lists can be defined for different terminals or sets of terminals. In Sample Program 2, one exit list is assumed; the address of this exit list is provided to VTAM in the EXLST operand of the ACB when the ACB is opened.

The following notes are keyed to Figure 5-29.

- 1 As in Sample Program 1, the ACB is opened and a SETLOGON is issued. A request to read input from any terminal is issued; the operation is performed asynchronously to the execution of the application program, and RPL exit-routine (RPL1) is designated for scheduling by VTAM when the operation completes. The terminal whose input is

read into Sample Program 2 is to be put into continue-specific mode. Thus, subsequent requests to read input from any terminal, issued in the RPL exit-routine, exclude the terminal whose input was just read and with whom the program is now in specific communication. The RECEIVE might be coded:

```
RECEIVE    RPL=RPLANY,AREA=AREAANY,AREALEN=100,  
           RTYPE=DFSYN,OPTCD=(ASY,ANY,CS)
```

RPLANY and AREAANY could be assembled in the program as fixed areas and reused each time the program issues a request to read input from any terminal. The ASY option of the OPTCD parameter specifies asynchronous request handling.

A branch is made to the wait routine, which waits for the first input to arrive from a connected terminal. The code in the initial routine is executed only once.

- 2 Both logical units and 3270 devices can be connected in the LOGON exit-routine. The INQUIRE macro instruction is used to determine which type of terminal is being connected. Storage that is to be associated with this terminal can be obtained from a pool or can be obtained dynamically from the operating system. (The storage can include an ECB, an RPL, and a work area for additional terminal-related information.) The address of this storage or other terminal-related information can be put in the user field of the NIB, using the MODCB macro; when the terminal is connected, VTAM saves the contents of this field and returns it to the program at the completion of each subsequent input from the terminal.

A logical unit or a 3270 might be connected as the result of a terminal-initiated logon request or as the result of an automatically-generated logon request (these logon possibilities are discussed in "VTAM Definition," in Chapter 3). A 3270 terminal-initiated logon request results from a terminal-operator action (pressing a function key or entering a logon message). A logical-unit logon request could be the result of either some terminal-operator action or could be initiated solely by the 3601 application program without involving a terminal operator (in either case, the actual request would be transmitted by the 3601 application program).

The following is a possible sequence of events that might occur prior to and during a logical unit terminal-initiated logon request:

- a. The 3601 and its logical units, defined to VTAM during VTAM definition, are made an active part of the VTAM network (perhaps by a network-operator VARY command).
- b. As a result of receiving an activation request for the 3601 controller, VTAM sends an activate-physical signal to the 3601. The 3601 acknowledges the signal and responds that it is ready for operation.
- c. The 3601 application program can either wait for a terminal operator at a 3601 device to indicate that the logical unit (work station) is to be logged on, or can issue a logon request on its own initiative. This is done by sending an initiate-session control signal to VTAM accompanied by the name of the VTAM application program with which the logical unit is to be connected (if standard logons can be specified by the logical unit) or accompanied by an installation-defined logon message.
- d. VTAM, receiving the initiate-session signal, schedules Sample Program 2's LOGON exit-routine.
- e. In the LOGON exit-routine, after the validity of the logon request is confirmed, the logical unit is connected, via an OPNDST macro instruction. The OPNDST causes VTAM to send a bind control signal to the 3601. When VTAM receives a response to the bind signal, it issues a start-data-traffic (SDT) indicator to the logical unit

(assuming SDT=SYSTEM is specified in the NIB used for connection). On receipt of a response to the SDT signal, the logical unit is connected to the application program, and the OPNDST is posted complete. (If SDT=APPL is specified in the NIB used for connection, the VTAM application program can itself send the initial SDT, using the SESSIONC macro instruction.)

The OPNDST might be specified for synchronous or for asynchronous request handling. If the latter, the LOGON exit-routine can either identify an ECB to be posted or an RPL exit-routine to be scheduled as soon as the connection has been made.

The same RPL and NIB could be used for each terminal being connected in the LOGON exit-routine.

It may be desirable to write an initial message to the connected terminal; this could be done from the LOGON exit-routine or in an RPL exit-routine following an asynchronously scheduled OPNDST.

As soon as the terminal is connected and control returned to VTAM, initial input for the terminal can cause scheduling of the RPL exit-routine, RPL1.

- 3 As soon as the first input is received from a connected terminal, the operation started by the request to read input from any terminal, issued in the initial routine, is completed. RPL1 is then scheduled and entered.

RPL1 can use the user field of the RPL of the request just completed to locate the terminal-unique storage and the identity of the terminal. On entry to RPL1, the address of the RPL is in register 1.

A CHECK macro is required to free the RPL for reuse; it also causes LERAD or SYNAD exit-routines to be entered if any error occurs.

RPL1 posts an ECB so that subsequently, the wait routine in the main program can determine that the input has been received and can pass the input to the processing routine analyzer. The RPL1 exit-routine moves input to the unique input-area for the terminal. It then reissues a request to read input from any terminal. Because a terminal-related RPL obtained in the LOGON exit-routine is used for subsequent I/O with any terminal just read, the RPL causing entry to RPL1 can be continuously reused by the RECEIVE in RPL1. The operation is to be asynchronous with relation to the program, and RPL1 is to be reentered each time the request is completed. Note that the terminal whose input caused entry to RPL1 is now in continue-specific (CS) mode. The RECEIVE would be coded identically to the RECEIVE in the initial routine.

Although not shown in Figure 5-29, the RPL exit-routine would send a positive response to the message that caused it to be entered if a positive response were requested by the logical unit. If a response was to be sent only on an exception condition, sending of an exception response would probably be performed in the SYNAD exit-routine after a CHECK was issued.

An alternative to having an RPL exit-routine for the RECEIVE with OPTCD=ANY and related logic is to have this logic located in the main program and have an ECB posted rather than an RPL exit-routine scheduled. In Sample Program 2, one advantage to using an RPL exit-routine is that input resulting from a RECEIVE with OPTCD=ANY is handled sooner in an RPL exit-routine than if an ECB were to be posted (which would require waiting until the next entry to the main program's wait

routine). This gives some preference to the first input of a new transaction or conversation over transactions or conversations already in progress.

- 4 The wait routine first determines whether an ECB has been posted, and if it has not, issues a WAIT macro on the list of ECBs. When a posted ECB is found, the RPL associated with it is located (perhaps by having the RPL located at a fixed displacement from the ECB), and a CHECK macro instruction is issued. The CHECK clears the RPL for reuse for the next VTAM request for that terminal and, if necessary, causes the LERAD or SYNAD exit-routine to be entered. On return from CHECK, the feedback fields of the RPL contain information provided by VTAM; in addition, the LERAD or SYNAD routine may have indicated action to be taken. If the operation was successful, the wait routine branches to an address associated with the ECB. In the case of the first input of a transaction or conversation, this address is that of the processing routine analyzer.
- 5 The processing routine analyzer, which might consist of separate routines for different types of terminals, analyzes the input and branches to or calls the appropriate processor. This processor may be in a higher level language, such as COBOL or PL/1. (These processor routines are depicted in 2 of Figure 5-1).
- 6 The processing routine processes the input and prepares the output. This may require one or more disk I/O operations, which might be performed by calling a common disk I/O routine. When output is ready, or, in a conversation, when the next input is required, the processing routine requests VTAM I/O, causing control to pass to an appropriate telecommunications I/O routine.
- 7 The disk I/O routine requests a disk I/O operation asynchronously and uses the wait routine to wait for completion. This allows processing for other terminals to continue while a disk I/O operation for one terminal is under way.
- 8 Although not shown, a processing routine might return to the next sequential instruction in the main program from which it was called; a branch would then be made to a common I/O routine, which would then branch to a 3600 or a 3270 input or output routine. A special routine might be required to edit 3270 input and format 3270 output.

If the terminal is a logical unit, an input or an output operation is requested as appropriate. The handling of the request is specified as asynchronous; completion is determined when the ECB related to the terminal is posted. Before issuing the request, the address to which the wait routine should branch (the return address is the processing routine) is placed in the ECB-related terminal block.

If input is requested, the input, when it arrives, is not used to satisfy the outstanding RECEIVE request in RPL1 because the terminal is in CS mode.

If output is requested, the data may be sent in a chain of transmissions. This might be useful with output that would be passed from the 3601 application program to a 3610 printer. The 3601 application program could store all elements of the chain in a buffer until the entire chain was received (or print each element as it arrived). The VTAM application program would ensure arrival of the entire chain by receiving a single positive response sent back by the 3601 application program when the last element of the chain was received.

If the output completes a transaction or conversation, the terminal is reset to continue-any (CA) mode so that input that begins the next transaction or conversation satisfies the RECEIVE with OPTCD=ANY request that is issued in RPL1.

Details of the 3600 routines are provided in Figures 5-30 and 5-31 and in accompanying notes.

- 9 Although record-mode (SEND and RECEIVE) is used to communicate with both types of terminals in Sample Program 2, the different characteristics of the terminal types may require different I/O routines. The size of I/O areas required may be different. The range of input that may arrive may be wider. An additional requirement may be to use brackets to control the overlap of input and output with 3270.

Details of the 3270 I/O routine are provided in Figure 5-32 and in accompanying notes.

- 10 The RESP exit-routine is scheduled and entered when a response arrives from a 3270 or a logical unit. A response is received by VTAM because the VTAM application program requested it in the RESPOND operand of the SEND macro. When the response is received, the operation, only posted as scheduled in the 3270 or the 3600 I/O routine, is now complete and the RESP exit-routine can now post the ECB. If the operation was successful, the response is positive; if an error occurred, an exception response will be indicated in the RPL. After copying the contents of the RPL (which is read-only) a CHECK macro is issued, causing the program's SYNAD exit-routine to analyze the exception and take possible action. The ECB is posted and control is returned to VTAM.

Details of the RESP exit-routine are in Figure 5-33 and accompanying notes.

- 11 In Sample Program 2, two kinds of asynchronous-flow indicators can be received from a 3601 application program: a request to stop sending to the terminal at the end of the chain that is currently being sent (a quiesce-at-end-of-chain (QEC) indicator) and a request to reinitiate sending after previously being requested to stop (a release-quiesce (RELQ) indicator). This use of these indicators may be desirable if an operator wants to interrupt a long series of printing so that keyboard input can be entered. After handling the operator request, the VTAM application program could resume printing. When either of these requests is received, VTAM schedules Sample Program 2's DFASY exit-routine.

This exit-routine does not apply to 3270 operation. Details of this routine are shown in Figure 5-34 and accompanying notes.

- 12 The TPEND exit-routine is scheduled and entered as the result of a request from the network operator for an end-of-day (normal) or quick closedown of the VTAM network. In addition to other possible processing, the TPEND exit-routine posts a special close ECB so that, subsequently, the main program's wait routine can branch to a CLOSE macro instruction in the main program.

- 13 The LERAD, SYNAD, and LOSTERM exit-routines handle different categories of errors or unusual situations. The LERAD or SYNAD exit-routine are entered as the result of a CHECK macro or a synchronous I/O request. The LOSTERM exit-routine is scheduled asynchronously when certain situations occur, such as the immediate deactivation of a connected terminal by the network operator. Like other parts of the program, the LOSTERM exit-routine might branch to the LERAD or SYNAD exit-routines for problem analysis. The LERAD exit-routine primarily handles logical errors; it is most likely that these would occur during the debugging stages of the program. This exit-routine might gather information, format it, and save it for programmer analysis after the program ends. The SYNAD exit-routine primarily handles physical errors; it determines what general action should be taken (for example, retry, disconnection of the terminal, termination of the program, or sending

a message to the network operator) and either takes the action or passes an action code to the main program or other exit-routine where the action is taken. The SYNAD routine may also want to record information related to situations that it handles for later problem analysis.

For record-mode terminals, a number of error situations must be perceived and analyzed as the result of receiving a response from a terminal; the response is analyzed following a RECEIVE with RTYPE=RESP specified or after a RESP exit-routine is entered. Errors or special situations that result in exception responses cause the SYNAD exit-routine to be entered when a CHECK macro is issued; the SYNAD can determine the cause of the exception response by analyzing sense information in the RPL and then take appropriate action.

### The Logic of the 3600 I/O Routine

This routine is entered directly or indirectly (perhaps from a common I/O branching routine in the main program) as the result of a request for input or output with a specific logical unit with which a processor is currently engaged in a transaction or a conversation.

Figure 5-30 shows the logic of the 3600 I/O routine. The following notes are keyed to this figure.

1 If the processor's request is for input, the information that must be passed to VTAM is set up, and a RECEIVE is issued. The address of the terminal's RPL is put in a register and the address of the input area associated with the logical unit and the length of the area are put in other registers. Since data is to be read, RTYPE=DFSYN is specified. The operation is to be asynchronous, and input is to be read only from the specific terminal (whose CID is located in the RPL's ARG field). The ECB associated with the logical unit is specified for posting by VTAM when the operation completes. After issuing the RECEIVE request, register 15 contains 0 if the request is successfully accepted, or some other return code if it was not. If the request is accepted, the wait routine is returned to, after setting the next sequential instruction in this routine as the address of the instruction to be branched to when the ECB is posted.

*Note:* For simplicity, most checks of register 15 are not shown in Sample Program 2.

2 When data is received from the logical unit, VTAM posts the ECB. When the wait routine detects the posted ECB, it branches to the indicated location in the 3600 I/O routine. The RESPOND field of the RPL can be tested to determine whether the logical unit wants a positive response returned so that it knows positively that the input was received. If so, a SEND is issued, indicating that a response is to be sent to the terminal (STYPE=RESP).

The SEND that sends the response, if requested, is scheduled synchronously. VTAM assumes POST=SCHED. Since no response can be returned to a response, once the request to send the response is accepted, the VTAM application program considers the sending of the response as complete.

If input arrives unsuccessfully or out of sequence (indicating that some input was lost), VTAM completes the VTAM application program's input request with an indication that input arrived for which an exception response must be returned; no input is forwarded to the program. The VTAM application program sends an exception response. The input request should be reissued.

Although not shown, the VTAM application program could also return an exception response to input that was successfully received. This might be done where such a

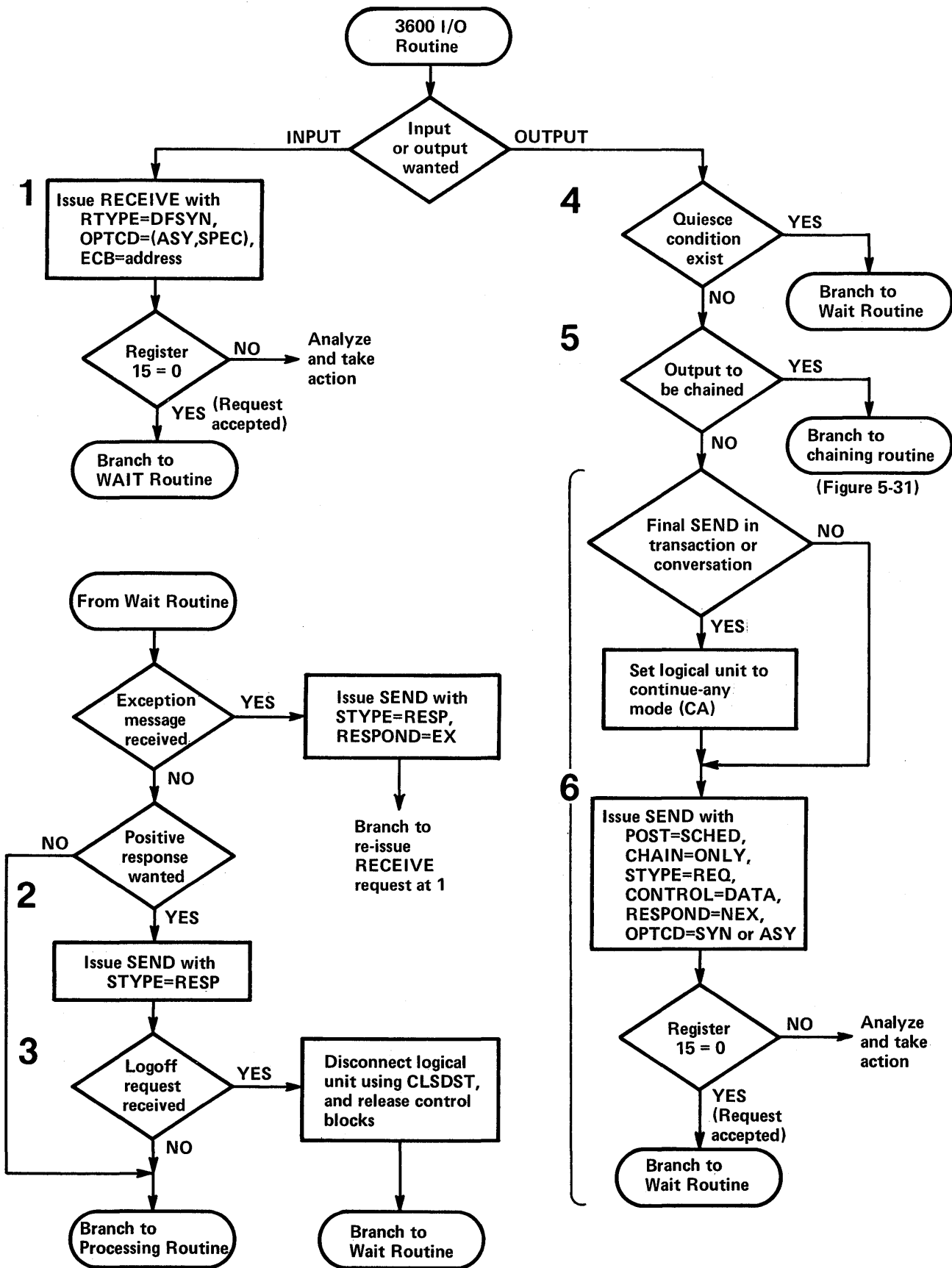


Figure 5-30. The Logic of the 3600 I/O Routine

response would have a meaning understood by both the VTAM application program and the logical unit. The USENSEO field of the RPL could be used to convey exception information.

- 3 If the input received contains a request to log off (to be disconnected from this program), the logical unit is disconnected via a CLSDST macro instruction, and the work areas associated with the logical unit are returned to the operating system or to a pool. Optionally, a message might be sent to the terminal, confirming logoff, prior to issuing CLSDST.

The above description of the 3600 I/O routine assumes that a CHECK macro is issued in the wait routine upon completion of each requested input operation; if an error occurs, CHECK causes entry to the LERAD or SYNAD exit-routine where appropriate analysis and action is taken. It might be noted that the input routine could issue a request to receive any kind of input: data or synchronous-flow indicators (DFSYN), asynchronous-flow indicators (DFASY), or responses (RESP). In this sample program, DFASY- and RESP-type input is handled by VTAM-scheduled DFASY and RESP exit-routines, but the logic in these routines could have been branched to after determining in the wait routine or in the 3600 routine that DFASY or RESP information had been received. DFSYN means that either data or synchronous-flow indicators can be received; although not shown in this example, synchronous-flow indicators might be received in some applications, in which case, such indicators would have to be responded to appropriately.

- 4 When an output request from a processor is received by the 3600 I/O routine, the I/O routine does not proceed with the request if the logical unit has quiesced the VTAM application program; instead, the I/O routine branches to the wait routine. The processor must wait for the quiesce to be released at which time the ECB for this logical unit is posted, a pending send request detected, and the I/O routine is reentered. (This logic is discussed in "The Logic of the DFASY Exit-Routine.")
- 5 If the output request is chained to other output requests, a branch is made to a chaining routine (see Figure 5-31).
- 6 If output is not being chained, a SEND is issued that includes the operand CHAIN=ONLY. If the output completes a transaction or conversation, the logical unit is returned to continue-any mode; its next input satisfies the RECEIVE (with OPTCD=ANY) request issued in RPL1. The request can specify scheduling of the operation (POST=SCHED) with completion to be determined as the result of a positive or exception response (RESPOND=NEX) that causes scheduling of the RESP exit-routine. (The RESP exit-routine posts the ECB associated with the terminal, thus notifying the VTAM application program and the processor that the output request was completed.) The 3600 I/O routine then branches to the wait routine.

### The Logic of the 3600 Chaining-Output Routine

Figure 5-31 shows the logic of the 3600 chaining output routine. The following notes are keyed to this figure.

- 1 Depending on the application, the number of elements in the chain might vary, or it might always be the same. Assuming that it varies in Sample Program 2, the number of chain elements must be determined so that the routine knows when to send the last element. It might be convenient to picture this routine as being entered as the result of a processor wanting to send a report to an administrative line printer; this report may vary in length between 20 and 100 printer lines. Each line is sent to the logical unit as a chain element. The logical unit would determine how many lines (chain elements) it would collect before forwarding them to the administrative printer.



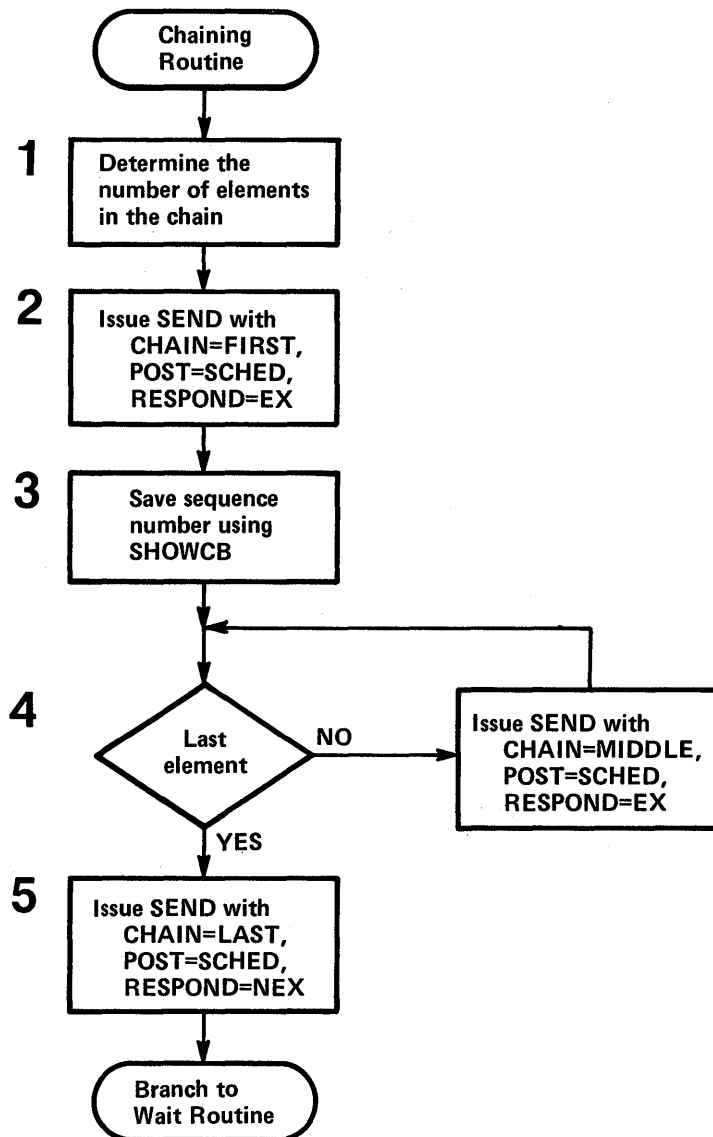


Figure 5-31. The Logic of the 3600 Chaining-Output Routine

This chaining routine might be passed all of the data to be sent in a chain or only part of it. In other words, the routine would not necessarily be sending an entire chain each time it was entered. The logic discussed here, however, assumes all, or, in a retry situation, the last part of a chain is being sent.

- 2 Because one of the advantages of chaining output is to reduce the number of required responses while still breaking output into segments that can be interspersed on the communication line, all SEND macros other than the last one specify that a response is to be returned only if an exception is noted (RESPOND=EX). When the last segment is received, a positive response is returned, and the VTAM application program recognizes that the entire chain arrived successfully. When RESPOND=EX is specified, the scheduling of output (POST=SCHED) is assumed by VTAM; it does not have to be specified.
- 3 In some cases, it may be necessary to save the sequence number of the first element sent in a chain. This number is available as soon as sending has been scheduled. It can be obtained from the SEQNO field of the RPL by using the SHOWCB macro. In case

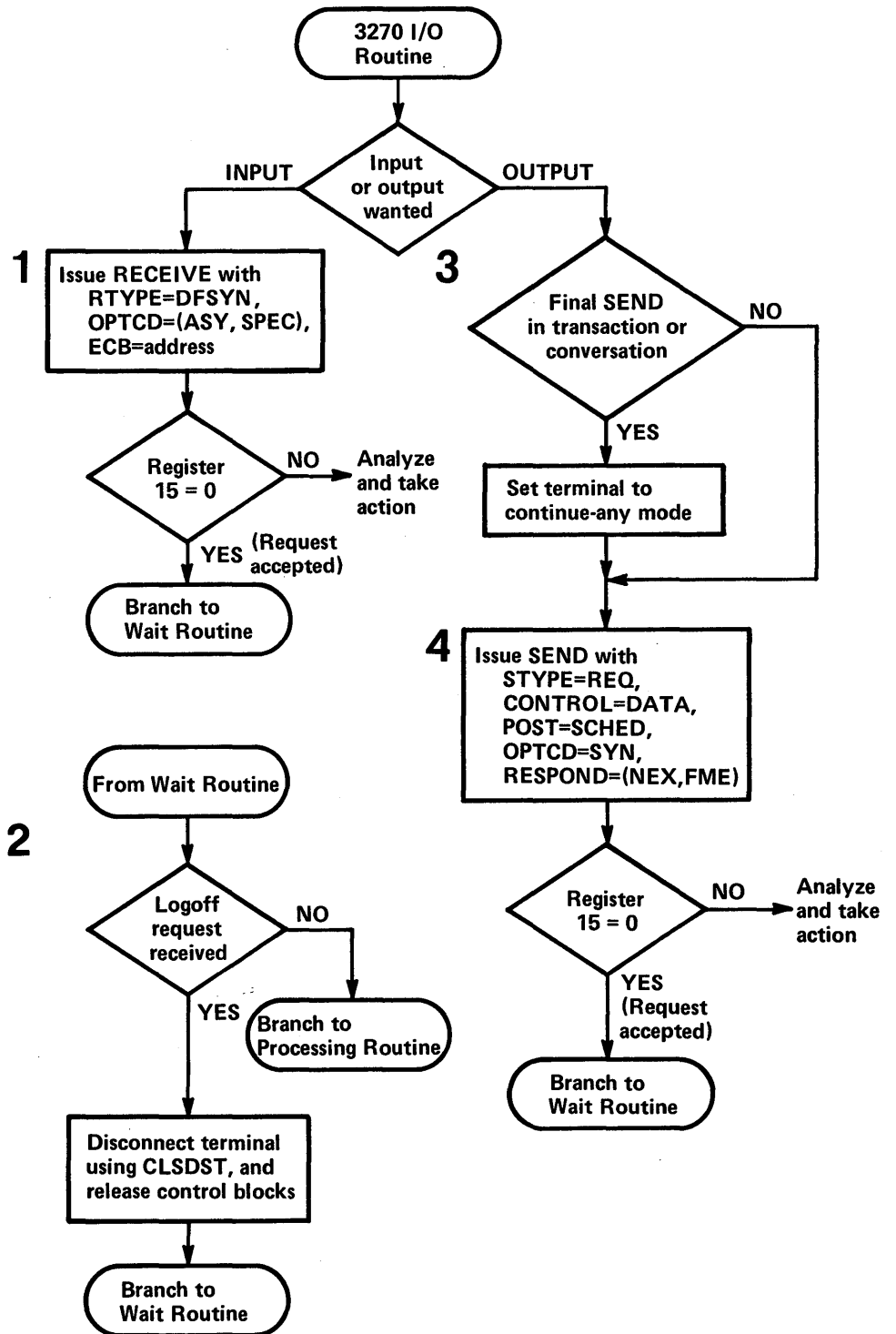


Figure 5-32. The Logic of the 3270 I/O Routine

all or part of the chain must be resent (an exception response arrives in the RESP exit-routine), the first-in-chain sequence number may be useful in determining where to start resending from. It may also be necessary (not shown here) to reset the beginning sequence number for the logical unit that is receiving the chain; this number would be sent to the terminal by using a SESSIONC macro. The sequence number can be saved in the work area associated with the terminal.

- 4 All elements except the first and last are middle elements (CHAIN=MIDDLE).
- 5 For the last element in the chain, the SEND macro must identify it as the last (CHAIN=LAST) and request the return of a response (RESPOND=NEX). Either POST=SCHED or POST=RESP can be specified.

When the response is received, if POST=SCHED is specified, the RESP exit-routine is scheduled; it posts an ECB, causing the wait routine to return to the processor that originated the output request. If POST=RESP is specified, VTAM posts an ECB or schedules an RPL exit-routine as appropriate.

### The Logic of the 3270 I/O Routine

Figure 5-32 shows the logic of Sample Program 2's 3270 I/O routine. With few exceptions, the VTAM application program need not distinguish between locally attached and remotely attached 3270s. Data received from a 3270 begins with an AID (Attention Identifier) character. Data sent to the 3270, whether local or remote, must begin with a 3270 command character (for example, an Erase, Erase and Write, or Erase All Unprotected character); VTAM inserts an ESC character for the BSC 3270. The 3270 is different from the logical unit in the following ways:

- Since a 3270 does not contain an application program (a variable program), the VTAM application program cannot exchange record mode indicators or responses with the 3270. However, in some cases, VTAM will provide responses to the VTAM application program on behalf of the 3270 as a result of receiving BSC responses to transmitted data or as a result of receiving indications that the 3270 is in a particular bracket state.
- The amount of data that can be sent to or received from the 3270 is limited by the physical characteristics of the 3270, whereas the amount of data that can be sent to or received from a logical unit is more indefinite.
- Chaining output to the 3270 is not possible.

The following notes are keyed to Figure 5-32.

- 1 Except that the type and length of data may be different for a 3270 RECEIVE, this request is similar to 1 discussed for the 3600 I/O routine.
- 2 This logic is similar to that of 2 for the 3600 I/O routine. However, because the 3270 cannot request that a response be sent to input it has provided, no check is made to determine whether to send a response.
- 3 If 3270 output is requested by a processing routine, the 3270 I/O routine determines whether this output completes a transaction or a conversation. If it does, the 3270 terminal is put back into continue-any mode so that the RECEIVE (with OPTCD=ANY) specified in the RPL1 exit-routine can receive input from this terminal when the terminal operator wishes to begin a new transaction or conversation.
- 4 A SEND macro instruction is issued to send the output. (Although not shown, this routine may also have to determine from the processing-routine request what 3270 command character—for example, Erase and Write—is to precede the output data

stream that the processing routine furnishes.) The sending of the output is scheduled synchronously (POST=SCHED,OPTCD=SYN); VTAM returns control after it has scheduled the output operation. A response is requested (RESPOND=(NEX,FME)) so that the VTAM application program can determine whether or not the operation was successful. The 3270 returns information enabling VTAM to provide the appropriate response in the RPL and to schedule the RESP exit-routine. The RESP exit-routine (Figure 5-33) posts an ECB so that the main program's wait routine can determine that the operation completed, branching back to the processing routine that requested the output. (Note that an output request to a 3270 printer requires a definite response (RESPOND=(NEX,FME)); an output request to a display can specify either NEX or EX but cannot specify that either a normal or exception response is to be returned (RESPOND=(NEX,NFME)).

After successfully scheduling output to the 3270, the 3270 I/O routine branches to the wait routine.

### The Logic of the RESP Exit-Routine

Figure 5-33 shows the logic of the RESP exit-routine. This routine is entered when the response is received to an output request that has POST=SCHED specified in a 3600 or 3270 I/O routine. The output operations have been scheduled with responses to be returned by the terminal so that completion of each operation can be determined. (It would also have been possible for all output operations to be specified POST=RESP; in this case, the response would be received by VTAM and its nature determined by the VTAM application program after ECB posting or RPL exit scheduling.)

Note that when the VTAM application program gets control in its RESP exit-routine, a RECEIVE is not issued. The nature of the response is determined by examining the RESPOND and other fields of an RPL that is in VTAM's storage (and therefore is read-only). The address of this RPL is pointed to in a parameter list whose address is in register 1 when the RESP exit-routine is entered. The terminal control area (the ECB, the RPL associated with the terminal, and the terminal-associated work area) can be located by the address in the USER field of the VTAM RPL. (It contains whatever was placed in the USERFLD field of the NIB when the terminal was connected.)

The following notes are keyed to Figure 5-33.

- 1 If the response was positive, the appropriate ECB is posted and a return is made to VTAM. Even if other action must be taken because the response is an exception, the ECB is posted so that the wait routine can determine that the operation has been completed.
- 2 The RESP exit-routine may wish to use the SYNAD exit-routine to analyze an exception response; if so, a CHECK can be issued and further action determined. (This would obviate the need to issue a CHECK in the wait routine for this operation.) Before issuing the CHECK, the contents of relevant fields of the read-only RPL must be moved to an RPL in the program since CHECK and other RPL-based macro instructions require that the RPL be modifiable. Alternatively, the user could set up the appropriate registers and branch directly to his SYNAD exit-routine.
- 3 If the situation was defined by the SYNAD exit-routine as recoverable, the operation would be retried. If it were part of a 3600 chaining operation, it might be necessary to save the sequence number of the output to which an exception response was returned so that the chaining routine could determine the sequence number at which it should start a retry. If a logical unit's inbound sequence number must be reset, a SESSIONC using the STSN operand could be used to synchronize sequence numbers.

On completion, the RESP exit-routine returns control to VTAM.

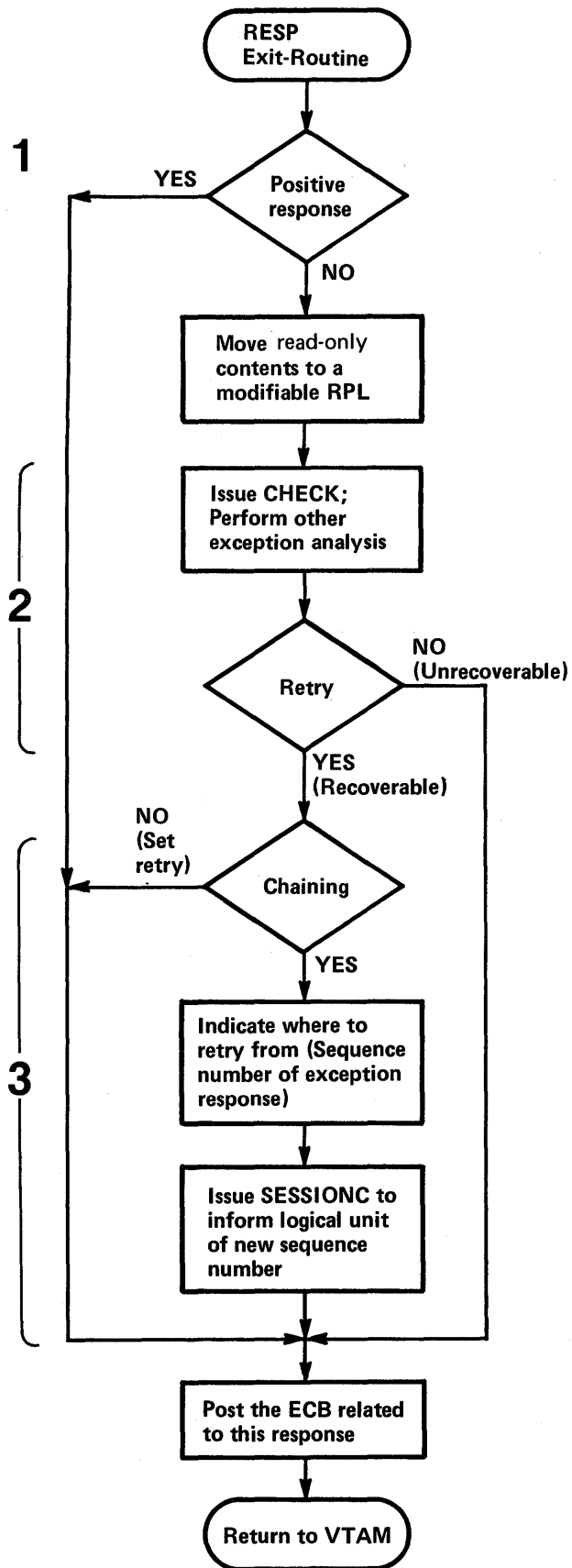


Figure 5-33. The Logic of the RESP Exit-Routine

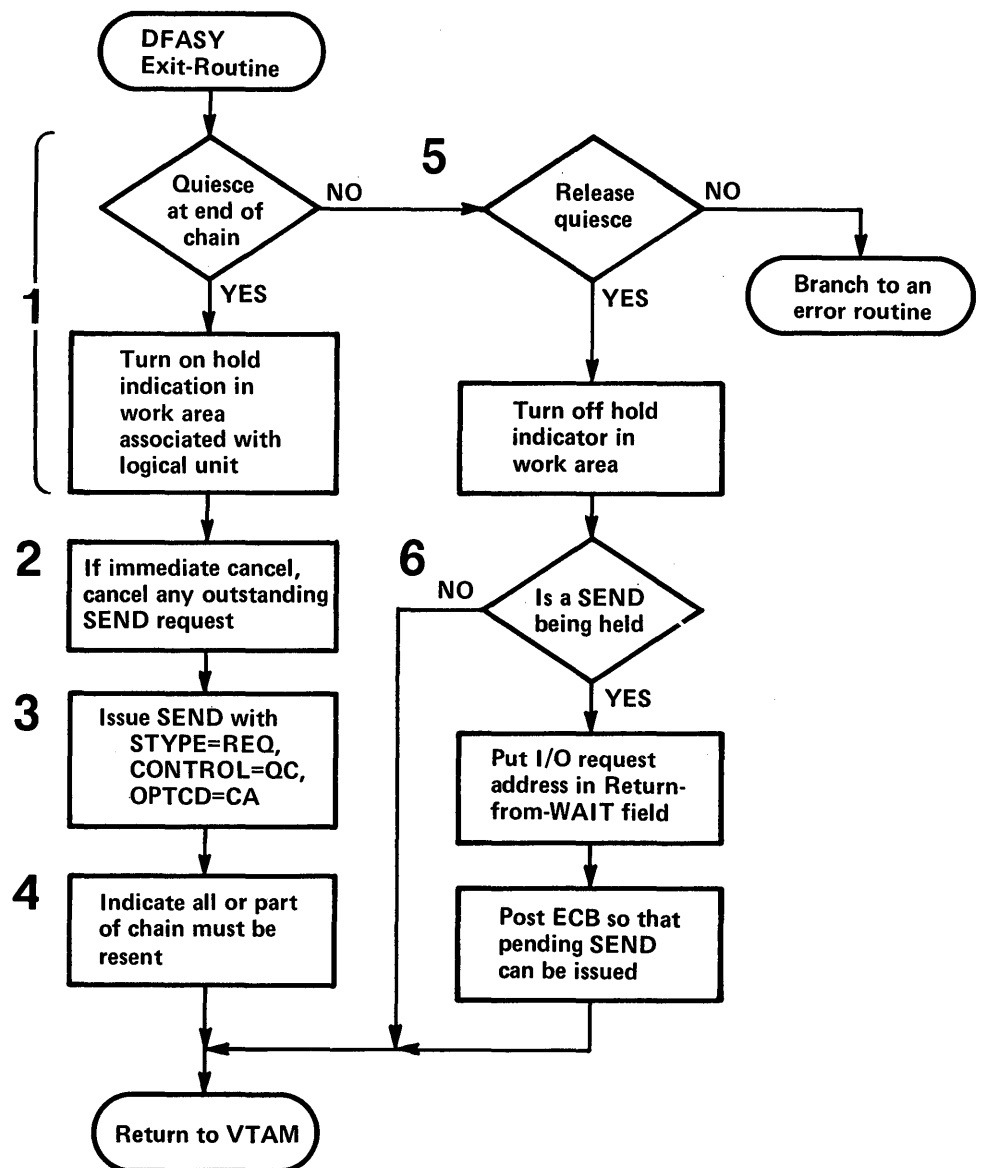


Figure 5-34. The Logic of the DFASY Exit-Routine

### The Logic of the DFASY Exit Routine

Figure 5-34 shows the logic of the DFASY exit-routine in Sample Program 2. The DFASY exit-routine is entered when a request is received from the logical unit to quiesce the sending of data to the logical unit or to resume sending it, if sending was previously quiesced.

Quiescing can be done for two reasons:

- To ensure that, at a given time, only the logical unit or the VTAM application program can be sending. This use of quiescing is not demonstrated in this sample program. (Quiescing is only one means available to ensure that both sides do not send at the same time. Change-direction indicators may also be used. In many cases, dependence on one end of the communication link receiving an answer to a previous output is sufficient to ensure that both ends do not send at the same time.)
- To interrupt a steady flow of data input so that an output operation can be performed. This is the use of quiescing that is demonstrated here. For example, a teller at a 3600 device may wish to temporarily interrupt a long printout so that an inquiry

can be submitted to the VTAM application program. As a result of a teller action, the logical unit for the teller's device sends a quiesce indicator to the VTAM application program, which can then agree to hold its sending in abeyance and be ready to read input from the logical unit.

The quiesce-at-end-of-chain and release-quiesce indicators are sent as asynchronous-flow messages unaccompanied by data. VTAM schedules the VTAM application program's DFASY exit-routine when one of these indicators is received.

The following notes are keyed to Figure 5-34.

- 1 The type of indicator that caused the DFASY exit-routine to be entered is available in the CONTROL field of the read-only RPL whose address is provided by VTAM on entry. If a quiesce-at-end-of-chain (QEC) indicator has been received, this routine sets a bit that prevents the sending of messages in the work area associated with the logical unit. The work area, as in the RESP exit-routine, is located by the address in the user field of the RPL.
- 2 If the quiesce is to be immediate, all scheduled but incomplete output operations can be canceled by clearing all existing data flow with a SESSIONC macro instruction that specifies CONTROL=CLEAR followed by a SESSIONC that specifies CONTROL=SDT. If in the middle of a chain and not all of the chain is to be resent, it may be necessary to determine where sending is to resume when the quiesce condition is released.
- 3 The QEC indicator is acknowledged by sending back a quiesce-completed (QC) indicator. The indicator is specified symbolically (CONTROL=QC) by the VTAM application program. So that the logical unit's synchronous-flow input (RTYPE=DFSYN) can complete the request to receive input from any logical unit, being recurrently issued in the RPL1 exit-routine, the logical unit is put back into continue-any mode (OPTCD=CA). In sending other than data, a response should not be specified; VTAM does not post the operation complete until it has received a response (in other words, POST=RESP and RESPOND=(NEX,FME) are assumed).
- 4 This flag may be required in addition to the hold-from-sending flag to determine where to resume sending. See 2 above.
- 5 If the indicator was a release-quiesce (RELQ), the hold-from-sending flag is turned off.
- 6 If further output is being held, the address of the output request routine is rescheduled for this logical unit and an ECB is posted so that the wait routine can branch to it. Control is returned to VTAM.

### ***Choosing Programming Alternatives Discussed in the Sample Programs***

The following is a summary of some alternatives to programming techniques discussed in the sample programs. In general, an installation must decide whether programming convenience or efficiency is more important. To achieve maximum efficiency, it may be necessary to "tune" a program; for example, to vary the number of consecutive RECEIVE with OPTCD=ANYs to determine the effect on response time.

- A RECEIVE with OPTCD=ANY that specifies ECB posting is more efficient than a RECEIVE with OPTCD=ANY that specifies the scheduling of an RPL exit-routine (which might then post an ECB). However, the RPL exit-routine provides some programming convenience and allows handling of an event's completion sooner than does the posting of an ECB by VTAM (which the program may not immediately detect).

- The use of more than one RECEIVE with OPTCD=ANY specified can be more efficient than a single RECEIVE with OPTCD=ANY. During the time between a single RECEIVE with OPTCD=ANY completing and being reissued, VTAM sends any input received to a pageable buffer from which the input must be retrieved when the RECEIVE is reissued. Multiple RECEIVES require multiple RPLs and input areas, however.
- If a positive response is required to confirm the arrival of an output message, a SEND with POST=RESP specified is more efficient than the combination of a SEND with POST=SCHED and the scheduling of a RESP exit-routine or use of a RECEIVE RTYPE=RESP.
- A SEND that specifies ECB posting is more efficient than a SEND that specifies the scheduling of an RPL exit-routine (in which an ECB may be posted). However, the latter may be easier to program.





## CHAPTER 6. RELIABILITY, AVAILABILITY, SERVICEABILITY

This chapter describes what VTAM does to detect errors, to attempt to recover from them, and—when recovery is impossible—to record the conditions under which the error occurred. The chapter also describes debugging aids, dump programs and test programs available through VTAM to correct programming errors and machine malfunctions.

### VTAM's RAS Strategy

VTAM's reliability, availability, and serviceability (RAS) aids are intended to minimize the impact of programming errors and machine malfunctions on the telecommunication network. The general VTAM approach to RAS is as follows. VTAM attempts to detect problems and handle them before they become serious. If an error is encountered, VTAM attempts recovery while recording the error for possible future maintenance. If recovery is not possible, VTAM automatically provides error records and dumps to help identify the problem and its cause. For additional debugging aid, the installation can request traces and testing services from VTAM. If possible, VTAM avoids terminating application programs or closing down the telecommunication system when an error occurs; instead, it tries to isolate the problem and either correct the error itself or provide information to enable the user to correct it. VTAM handles both hardware and software errors; software errors include those on the part of users (application programs and the network operator), of the operating system, and of VTAM itself.

Error-recovery and error-recording operations are invoked automatically. VTAM requires no action on the part of the installation to invoke them, although the error records could be used to perform maintenance on elements in the system that are causing frequent temporary, but recoverable, errors.

When an error is permanent, a message is usually sent to the network-operator's console. The message and its explanation define the condition, indicate the probable cause, and suggest a course of action. For the network operator, this action may involve circumventing the problem as well as collecting data for later debugging. To get adequate material for problem determination, the network does one or more of the following things:

- Saves all console logs that pertain to the error message. These logs would probably reflect all of VTAM's actions from VTAM startup through the issuing of the message.
- Obtains printouts of any requested traces performed in connection with the error.
- Saves any dumps that may have resulted from the errors.
- Requests and saves status displays of the nodes involved in the error.
- Initiates the Teleprocessing Online Test Executive Program (TOLTEP) for testing devices and lines involved in the error.

As a temporary solution to a problem, the network operator may be able to circumvent it or avoid it by disabling part or all of the telecommunication system. In any case, to assist in later correction of the problem, the installation should keep a complete description of the network as it existed at the time of the error.

Correcting the problem involves identifying the cause and then making the necessary correction. The steps involved in identifying the problem include:

1. Studying the message log—The sequence of messages leading up to the error as well as the error message itself may identify the problem.

2. Examining error records—Error records pertaining to VTAM or its telecommunication system may identify the problem.
3. Examining dumps and traces—Dumps and traces can be used to identify the area in which the error occurred or to find the problem itself.
4. Recreating the problem—VTAM's network-operator commands and RAS facilities can be used to help recreate the problem and to collect pertinent data.

## VTAM's RAS Facilities

VTAM's RAS facilities can be grouped into those related to serviceability and those related to reliability and availability of the system. The following sections detail VTAM's RAS facilities.

### *Serviceability Aids*

VTAM's serviceability aids assist in determining the causes of problems in the telecommunication system. Serviceability aids include:

- **Error Recording**

*Hardware Errors:* VTAM's hardware-error recording augments the error recording of the operating system. Both permanent and temporary errors are recorded. In addition, messages are sent to the network operator when any permanent error occurs.

*Software Errors (OS/VS):* VTAM uses SYS1.LOGREC to retain records of software errors that result in the abnormal termination of VTAM.
- **Traces**

*DOS/VS:* The problem determination and serviceability aids (PDAIDS) can be used to monitor VTAM's SVCs and I/O operations.

*OS/VS:* The generalized trace facility (GTF) can be used to monitor VTAM's SVCs, I/O operations, and task management and to trace events in application programs using VTAM.

*VTAM:* Internal VTAM traces can be used to monitor I/O activity and to record contents of the buffer within VTAM.
- **Dumps**

*OS/VS:* System dump programs are tailored by VTAM to provide formatted dumps of some VTAM control blocks and in some cases, to provide additional diagnostic information.

*NCP:* VTAM has tailored the NCP dump utility program. This utility program can be invoked using VTAM's MODIFY command, or, optionally, it can be invoked as part of VTAM's error recovery procedures for communications controllers. The dump provided is of the NCP in the communications controller.
- **TOLTEP**

The Teleprocessing Online Test Executive Program (TOLTEP) provides online testing for telecommunication lines and certain devices in a VTAM system.

Each of these aids are discussed in detail below.

### **Error Recording**

This section describes VTAM's facilities for recording errors encountered in the telecommunication system. Hardware error recording and software error recording are discussed separately.

**Recording Hardware Errors:** Hardware error recording is a function of the error recovery procedures (ERPs). (See "Reliability and Availability Support," later in this chapter, for a description of ERP processing.) The data on hardware errors collected by VTAM is placed in the error-record data set of the operating system. The information in this data set can

be formatted and printed by each system's Environmental Recording, Editing and Printing (EREP) program.

VTAM's hardware error recording is an extension of the OS/VS Outboard Recorder and the DOS/VS Recovery Management Support Recorder.

In addition to recording errors, VTAM maintains two counters for each locally attached device in the network. One counter keeps track of the number of SIO commands issued for the device; the other keeps track of the number of temporary errors for the device. Counters are also kept in the device statistic tables that the operating system maintains for each locally attached device. These counters indicate the number of each type of error encountered for each device. The tallies in the counters are included in any records written to the error-record data set.

Recording occurs when any of the following conditions is encountered:

- *Permanent hardware error*: This is an error for which recovery could not be made, either because the error is undefined or because attempts by ERPs to correct the problem were unsuccessful.
- *Counter overflow*: A record is written whenever any of the counters is about to overflow.
- *End-of-day*: A record is written whenever a device is deactivated by VTAM.

Records for permanent errors include the name and address of the failing device, the date and time of the failure, the contents of the counters at the time of the failure, the failing channel command word, the channel status word, sense information, and device flags.

Records for counter overflow and end-of-day conditions include the name and address of the associated device, the time and date that the event occurred, and the contents of the counter.

**Recording Software Errors:** In OS/VS systems, if VTAM cannot recover from a software error and must terminate processing, VTAM collects data on the error. This data is formatted and written to the system data set SYS1.LOGREC. Records on this data set can be formatted and printed by the OS/VS EREP program.

A VTAM software error record includes a description of the error and audit-trail information. A VTAM audit trail is a record of the modules entered during the processing in which the error occurred. (An audit of module activity is maintained for those areas of VTAM responsible for processing telecommunication requests.) The audit information in the software error-record identifies the module executing when the error was detected, as well as the modules that were entered prior to the error.

## Traces

Trace facilities for VTAM under both DOS/VS and OS/VS is at two levels:

- Operating-system traces.
- VTAM traces (including NCP-provided traces).

The operating-system traces provide the highest level of problem determination. By using the appropriate system trace, a problem may be isolated to a particular program or operating-system component. If the problem is isolated to a component such as VTAM or to an application program using VTAM, the VTAM traces can then be used to help locate and identify the area in which the error is occurring.

Operating system traces and VTAM traces are each discussed in more detail below.

**Operating-System Traces:** The system traces can be used to monitor the SVC and the I/O activities of VTAM. The data collected by these traces can be used as a chronology of VTAM activity and reflects the conditions in VTAM during errors.

System traces for VTAM are started and stopped by using the facilities of the operating system. DOS/VS trace records for VTAM are printed with the PDLIST program. OS/VS1 trace records for VTAM are printed with the operating system's HMDPRDMP service aid. Refer to the publication *OS/VS1 Service Aids*, GC28-0665, for information on HMDPRDMP. OS/VS2 trace records for VTAM are printed using the operating system's AMDPRDMP service aid. Refer to the publication *OS/VS2 System Programming Library: Service Aids*, GC28-0633, for information on AMDPRDMP.

**VTAM Traces:** VTAM provides the following types of traces:

- *I/O traces*—To trace I/O activity within VTAM.
- *Buffer traces*—To record the contents of VTAM buffers as data enters and leaves VTAM.
- *NCP line traces*—To record the NCP trace records of specified line.

VTAM traces are initiated and terminated by VTAM start options or the MODIFY command. See "VTAM Commands," in Chapter 4, for information on starting and stopping VTAM traces.

The trace records generated by VTAM traces are written on a particular data set in each operating system. For DOS/VS, the file identification of this data set is TRFILE. For OS/VS, VTAM passes the trace data to GTF, which records the records in SYS1.TRACE. Because VTAM traces use GTF to record the trace records, GTF must be active before VTAM data can be recorded. If GTF is not active, VTAM does not provide a record of the event to be traced.

To print VTAM trace records, VTAM provides a utility program for DOS/VS and uses a system utility program for OS/VS. The VTAM trace-print utility program for DOS/VS selectively edits and prints the data collected by the various VTAM traces. Using the MODIFY command, the network operator starts the print utility program and specifies the nodes for which trace records are to be printed. In OS/VS1, VTAM traces are printed by using the operating system's HMDPRDMP service aid. In OS/VS2, VTAM traces are printed by using the operating system's AMDPRDMP service aid.

Refer to the publication *DOS/VS Serviceability Aids and Debugging Procedures*, GC33-5380, for more information on the PDAIDs and the PDLIST program. Refer to the publication *OS/VS1 Service Aids*, GC28-0665, for more information on GTF and the HMDPRDMP service aid. Refer to the publication *OS/VS2 System Programming Library: Service Aids*, GC28-0633, for more information on GTF and the AMDPRDMP service aid.

## Dumps

VTAM provides routines to augment both the operating system dump facilities and the NCP dump facilities. The text below describes what these routines do.

**Operating System Dumps:** VTAM provides routines to format portions of OS/VS dumps. When the operating system is dumping VTAM or an application program that is using VTAM, it invokes these routines to locate and format key VTAM control blocks. For each control block that is formatted, its name and address is printed, followed by the name and contents of the pertinent fields. A hexadecimal dump of the control block is printed following the formatted printout. If VTAM finds an invalid field (either in the control block or in the chain of pointers to the control block), the condition is noted in the dump.

This formatting is performed automatically for dumps resulting from abnormal termination (ABEND) of VTAM itself, or from the abnormal termination of an application program that is using VTAM, or from a SNAP macro instruction in an application program. For dumps produced by the PRDMP service aid, however, VTAM formatting is optional; the option must be specified as part of the input to the service aid.

Dumps of VTAM include audit trail information and unique identifiers for each VTAM module and control block. The audit-trail information is like that recorded for software information.

**NCP Dumps:** VTAM uses the NCP dump program to dump the contents of communications controllers. An NCP dump is taken automatically if the NCP fails and an automatic dump was specified as part of the generation of that NCP. If an NCP fails and an automatic dump was not specified, the network operator is notified of the failure and is given the option of requesting a dump of the NCP. Other than during error recovery, the network operator can also request a dump of an NCP by using the MODIFY command, as described in "Starting and Stopping VTAM Facilities," in Chapter 4.

**Teleprocessing Online  
Test Executive  
Program (TOLTEP)**

TOLTEP is a VTAM component that controls the selection, loading, and execution of Online Tests (OLTs) within the telecommunication network. These OLTs are specific device tests designed to be used to diagnose hardware problems and to verify the reliability of a device in the telecommunication network.

TOLTEP allows multiple OLTs to be run concurrently with processing application programs in the VTAM telecommunication system. TOLTEP conducts the testing of start-stop, BSC, and SDLC lines; it tests devices attached via start-stop or BSC lines; and it tests locally attached 3270s and 3272 control units. In OS/VS, TOLTEP can also be used to dynamically modify its configuration data sets (CDS files). Testing of SDLC teleprocessing subsystems is not conducted by VTAM; such testing is a function of the subsystems themselves.

TOLTEP is included automatically in the system when VTAM is generated. It requires that the OLT option in the NCP be specified during NCP generation. TOLTEP is automatically initialized and made ready to accept test requests when VTAM is started. If TOLTEP is abnormally terminated prior to VTAM termination, VTAM automatically attempts to restart it.

To run TOLTEP, a terminal must be available for use as a control terminal. This terminal could be the terminal to be tested, although this is usually unwise because it may be impossible to enter test data or receive test results at the terminal. The control terminal must be able to enter and receive alphameric characters. An alternate printer can be designated to receive hard-copy output from the test. (The alternate pointer is required if the control terminal is also the terminal being tested.)

TOLTEP can be invoked either from the network-operator's console or from start-stop or BSC terminal. As explained in "Starting and Stopping VTAM Facilities," in Chapter 4, TOLTEP can be invoked from the console via an option of the MODIFY command or by using the VARY command to log a specific terminal in the network onto TOLTEP.

If a terminal is being monitored by the network solicitor (and is, therefore, not connected to an application program), a terminal-initiated logon can be used to log the terminal on to TOLTEP.

If a terminal to be used by TOLTEP is currently connected to an application program, the connection must be broken before the terminal can be made available to TOLTEP.

VTAM allows an application program to be notified if a connected terminal is to be used by TOLTEP; although the application must issue a CLSDST macro instruction to release the terminal. If a connected terminal specified in a test request is not released by the application program, TOLTEP waits until the terminal is available before proceeding with the test. Thus, the installation should assure that application programs are designed to release terminals when needed by TOLTEP.

When TOLTEP is invoked, it requests information from the operator at the invoking terminal (or the terminal specified in a network-operator logon for TOLTEP). Requested information includes the name of the control terminal (unless a logon was used to invoke TOLTEP, in which case the terminal specified in the logon is used as the control terminal), the name of the alternate printer (optional), the name or names of the nodes to be tested, and the tests to be executed. If TOLTEP was invoked by other than the MODIFY command, the network operator is notified of the terminals and other nodes specified in the request to TOLTEP and must grant permission for the testing to proceed. Any additional nodes identified by the control terminal for testing must also be approved by the network operator. (The installation-coded authorization exit-routine must permit TOLTEP to connect and disconnect from terminals involved in tests.)

Once authorization has been given by the network operator, TOLTEP connects to the appropriate terminals and begins testing the specified nodes according to instructions from the control terminal and the appropriate OLTs.

### ***Reliability and Availability Support***

The purpose of VTAM's reliability and availability support is to maintain the operation of the telecommunication network. This support attempts to prevent problems, and if that is not possible, to minimize the impact of the problems. (To enhance reliability and availability, most of the VTAM modules are reusable or reenterable.) VTAM's reliability and availability support includes:

- **Error Detection and Feedback**  
Before attempting to act upon any request, VTAM analyzes it for any erroneous information. If an error is detected, VTAM returns the request along with an indication of the error.
- **NCP Initial Test**  
VTAM allows a communications controller to be tested before an NCP is loaded. This test can be used to identify problems in a controller before it is made an active part of the system. This testing is optional for locally attached communications controllers; it is required for remotely attached communications controllers.
- **Storage Management**  
VTAM controls the allocation of much of the storage required for its operation. Using this control, VTAM permits the queuing of requests and attempts to avoid storage interlocking conditions. (A storage interlocking condition is one in which so much of VTAM storage has been devoted to the initiation of request processing that not enough is available to complete that processing.)
- **Error Recovery**  
**Hardware Errors:** Using the facilities of the operating system and of the NCP, VTAM attempts to recover from some errors. If recovery is not possible, a permanent error is recorded and VTAM attempts to reallocate system resources to reduce the impact of the failure. If recovery is possible, processing continues, but a count is maintained of the initial failure.  
**Software Errors:** VTAM tries to recover from some errors. If recovery is not possible, VTAM first attempts to isolate the problem and continue processing. If VTAM cannot continue processing, it attempts to orderly close down the telecommunication system so as not to affect the non-telecommunication jobs in the host CPU.

- NCP slowdown  
VTAM recognizes NCP slowdown and assists the NCP to return to normal operations.
- Configuration Restart  
If an error in the telecommunication network causes an NCP to terminate, VTAM attempts to restart the affected NCP.

These operations are discussed in more detail below.

### **Error Detection and Feedback**

All requests from the network operator and from application programs are initially tested for errors. If an error is detected, the request and an indication of the error are returned to the requester. VTAM does not process a request it determines to be in error. If, during the processing of the request, an invalid condition is encountered, VTAM stops processing the request and returns an error indication to the requester. No further processing is done for the request.

If the requester is the network operator, VTAM first checks the syntax of the command. If there is an error in syntax, the command is rejected with a message to the network operator; the message describes the error and specifies where it occurred. If the syntax is valid, the requested function is validated. (For example, a request to activate a minor node whose major node is inactive is an invalid function request.) If the function requested is invalid, the command is returned to the operator along with an explanation of the error.

If both the syntax and the function are validated but VTAM encounters an error while processing an operator command, the network operator is notified of the unsuccessful operation and of the reason for the failure.

In addition to notifying the network operator of error conditions directly attributable to operator commands, VTAM transmits message to the console describing any unusual or error conditions that affect the operation of the telecommunication system; for example, permanent hardware errors and the abnormal termination by VTAM of an application program.

If an error condition is found in either an application program's request or in the processing of such a request, VTAM attempts to notify the application program of the error. Notification is made by scheduling a user-specified exit-routine or by a return code. VTAM has numerous return codes to help isolate the reason for the error. VTAM avoids abnormally terminating application programs if possible. Using the exit-routines and the return codes, the application program can attempt to correct the error, bypass the error, or terminate processing.

### **NCP Initial Test**

The NCP initial test is a facility of the communications controller. It is exercised automatically for a remotely attached communications controllers; it is an option for locally attached communications controllers. If testing is specified in the generation of an NCP for a locally attached communications controller, VTAM automatically invokes the test program prior to loading that NCP. If testing is not specified, initial testing of the NCP is not performed for a locally attached communications controller.

### **Storage Management**

VTAM has a number of storage pools, as noted in "VTAM Buffering," in Chapter 3. The installation can specify the size of each pool and a threshold value for each.

These storage pools are used to obtain buffers to transfer data between the host CPU and the network and to obtain storage for VTAM control blocks needed to service each telecommunication request. Because these pools are in VTAM storage and are managed



by VTAM, each application program need not be significantly concerned with providing storage in its own partition or address space for VTAM. (The application program need provide storage only for the work areas and VTAM control blocks, such as the ACB, EXLST, NIB, and RPL, that it uses directly.)

VTAM consolidates much of the storage requirements for buffers and control blocks for all of its application programs and enables the storage pools to be shared among these applications.

VTAM manages its storage pools by determining when storage should be requested, the amount of storage to request, and whether sufficient space is available to meet the request. VTAM also enables requests for buffers to be queued if sufficient space is not available to meet demands. By queuing storage requests, VTAM reduces the need to abnormally terminate an application program when insufficient storage is available at a particular time for its operation.

The pool used to satisfy a storage request within VTAM depends upon the type of request. For example, the pool used to obtain storage for data buffers is not the same as the one used for control blocks, and the pool used to obtain storage for a control block built during the initial stages of processing a request is not the pool used to obtain storage for a control block built during the completion of that request processing. By discriminating among storage requests, VTAM attempts to avoid storage interlocking conditions. If one pool should become temporarily exhausted, the VTAM procedures that need storage from that pool may be forced to wait until it is replenished, but procedures that need storage from other pools can continue processing.

In OS/VS, VTAM also allows the installation to specify a maximum threshold value for each pool. When a pool is operating above its threshold, only priority requests (determined by VTAM) for storage from that pool are satisfied. Use of the threshold option can assist VTAM to maintain the availability of the telecommunication system by allowing critical telecommunication functions to continue even if the noncritical functions must wait temporarily. When storage is not available in a pool, VTAM queues storage requests for that pool; then, when storage is available, those requests are serviced.

Establishing buffer limits for terminal/application-program connections is another option that affects VTAM's storage management. Using this option, the installation can prohibit low-priority jobs from monopolizing VTAM's storage pools.

In addition to managing storage by controlling its allocation, VTAM attempts to protect its storage from unauthorized access. VTAM components, control blocks (except for application programs' ACBs, NIBs, and RPLs), and buffers reside in storage protected from nonprivileged users by an operating-system key.

## **Error Recovery**

VTAM attempts to recover from both hardware and software errors. Both kinds of recovery are discussed below.

**Hardware ERPs:** When an I/O error interruption occurs, the appropriate error recovery procedures (ERPs) are invoked (within the operating system or VTAM for errors on locally attached devices and within the NCP for errors on remotely attached devices). The ERPs attempt to determine the type of error and to recover from it.

When a permanent error is encountered for a locally attached device, the ERP notifies the network operator of the problem and creates an error record as described in "Serviceability Aids," earlier in this chapter. When a temporary error is encountered, no message is sent to the console although a count is maintained of the temporary errors for that device.

Each NCP provides ERPs for the devices attached to it. When an NCP has completed error-recovery processing, it transmits an error record to VTAM for recording on the operating system's error-record data set. For a permanent error, a message describing the problem is sent to the network-operator's console.

**Processing Software Errors:** When VTAM encounters a software error, it attempts to determine whether the error resulted from action on the part of the user, the operating system, or VTAM itself. If it is a user error, it is handled as explained in "Error Detection and Feedback," earlier in this chapter.

If VTAM cannot determine whether the problem is a user error, the access method attempts to recover from the error. If recovery is not possible, VTAM attempts to isolate the cause of the problem and to bypass it. If this partial recovery is not possible, VTAM abnormally terminates, attempting to close down the telecommunication system without impacting the rest of the operating system. Note that if a problem can only be alleviated by abnormally terminating the application program associated with the problem, VTAM terminates the application. Terminating the application program is an attempt to reduce the impact of an error on the total telecommunication system; terminating one application program still permits other applications to continue to use the telecommunication system.

#### **NCP Slowdown**

If telecommunication activities overload an NCP and exhaust its buffers, the NCP automatically enters slowdown mode. In slowdown mode, the NCP reduces its acceptance of input data (from both the network and from the host CPU) and attempts to increase the rate of output. VTAM recognizes an NCP slowdown, and, to facilitate NCP recovery, VTAM stops scheduling input and output requests for that NCP, although VTAM does continue to read from the NCP. During this slowdown processing, VTAM continues to accept requests from application programs, but it queues those requests that are directed to nodes in the network controlled by the NCP in slowdown mode. When the NCP has recovered and is no longer in slowdown mode, the queued requests are processed and sent to the NCP.

#### **Configuration Restart**

VTAM provides the capability to restart an NCP in a communications controller after a failure occurs in that controller requiring the reloading of the control program. An NCP can be restarted only if the original failure was in the communications controller. If the failure is in the host CPU, configuration restart cannot be used.

When an error occurs in a communications controller, VTAM's ERPs determine whether the channel (for a local attachment) or the line (for a remote attachment) failed. If neither failed, VTAM initiates the NCP dump program (if automatic dumping was specified in the generation of the NCP or if requested by the network operator). NCP initial testing is performed if the communications controller is remotely attached or if the communications controller is locally attached and the NCP was generated with the test option requested. Next, if the network operator specifies the NCP is to be reloaded, VTAM attempts to reload and restart it. If the NCP is restarted, VTAM attempts to reinstate the network status at the time of the failure. That is, nodes active at the time of failure are reactivated (if possible), any remotely attached communications controllers active at the time of the failure are restarted if possible, any line-scheduling parameters modified by the network operator are reestablished, and (if the NCP included PEP) lines are reinstated to the mode they were in when the failure occurred.

If an NCP is successfully restarted, VTAM attempts to restore the connections between application programs and terminals:

- For application programs that were connected to terminals in basic-mode, the connections are maintained by VTAM for those terminals that were successfully

reactivated. Outstanding I/O requests for these terminals are purged, and the application programs must resubmit their requests.

- For application programs that were connected to terminals in record-mode, the application programs must issue CLSDST and OPNDST macro instructions (in that order) for the terminals.
- For application programs connected to terminals that could not be reactivated, the connections are lost; the applications should issue a CLSDST macro instruction for those terminals but can continue processing with unaffected connections.

## CHAPTER 7. VTAM PLANNING CONSIDERATIONS AND REQUIREMENTS

The first part of the chapter states the machine and program requirements (including NCP requirements) for using VTAM. The second part of the chapter discusses ways an installation can plan for and take advantage of features and facilities in VTAM.

### Machine Requirements

This section discusses the machine requirements for a telecommunications system with VTAM. It specifies which units can be used with VTAM including the central processing units (CPUs) and their required features, communications controller requirements, and terminals and terminal features.

#### *CPU Support*

VTAM is a System/370 access method and runs in a virtual storage environment on one of the following CPUs with the relocation feature:

- Under DOS/VS—System/370 Models 115 (using VTAM in only basic-mode), 125, 135, 145, 155II, 158, and 158 Submodel 2 (available for World Trade countries only; that is, only in countries outside of the United States).
- Under OS/VS1—System/370 Models 135, 145, 155II, 158, 158 Submodel 2 (available for World Trade countries only; that is, only in countries outside of the United States), 165II, and 168.
- Under OS/VS2—System/370 Models 145, 155II, 158, 158 Submodel 2 (available for World Trade countries only; that is, only in countries outside of the United States), 158MP, 165II, 168, and 168MP.

The host CPU must be equipped with the Compare and Swap and the Compare Double and Swap instructions. These instructions are available as follows:

- Optional on the System/370 Model 135 via the Conditional Swapping Feature 1051.
- Optional on the System/370 Model 145 via the Advanced Control Program Support Feature 1001 or via the Conditional Swapping Feature 1051.
- Standard on the System/370 Models 115, 125, 155II, 158, 158 Submodel 2 (available for World Trade countries only; that is, only in countries outside of the United States), 158MP, 165II, 168, and 168MP.

#### *Locally Attached 3270s*

VTAM provides communication with IBM 3270 Information Display Systems that are locally attached to the host CPU. The maximum number of 3270s that can be used and any required features are determined by the 3270 and the system, not by VTAM.

Appendix A lists the devices and features for locally attached 3270s that can be used with VTAM.

#### *Requirements for Communications Controllers*

For remotely attached devices, VTAM requires at least one locally attached IBM 3704 or 3705 Communications Controller.

VTAM uses only the network control program/VS (NCP) for communications controllers. VTAM uses only the network control mode of the NCP, and it uses this mode with or without the partitioned emulation programming (PEP) extension.

VTAM can use both the locally and the remotely attached communications controllers. A remote communications controller must run in network control mode only and must be

attached to a local communications controller via an SDLC line controlled by the network control mode of the NCP in that local controller.

An NCP used by VTAM requires at least 48K bytes of storage in the communications controller; therefore VTAM can use the following communications controller models:

- IBM 3704 Communications Controller—Models A3 and A4.
- IBM 3705 Communications Controller—Models A2, B2, B3, B4, C2, C3, C4, C5, C6, D2, D3, D4, D5, D6, D7, and D8.

VTAM does not support the following models because they have less than 48K bytes of storage:

- IBM 3704 Communications Controller—Models A1 and A2.
- IBM 3705 Communications Controller—Models A1, B1, C1, and D1.

Except for storage capacity, the features required on a communications controller do not depend upon VTAM. VTAM, because it uses the NCP, requires a communications controller with a minimum of 48K of storage. Required features depend upon the intended application of the communications controller (including local or remote attachment) and the type of control program to be used (NCP with or without PEP).

**Channel Adapter Support:** Support of channel adapters is a function of the communications controller and the NCP, and as such, is as follows:

- The locally attached 3704 is equipped with the Type 1 Channel Adapter.
- The locally attached 3705 can be equipped with either the Type 1, Type 2, or Type 3 Channel Adapters.

**3705 Two-Channel Support:** Using NCP facilities and a Type 3 Channel Adapter, VTAM supports the attachment of two channels to a 3705 in an OS/VS2 tightly-coupled multiprocessing environment. This attachment is not available for a loosely-coupled multiprocessing environment for two independent CPUs.

VTAM does not provide support for the manual Two-Channel Switch (8002) for the 3705. Support for this feature in a VTAM system is the responsibility of the installation.

**Other Features not Supported:** VTAM does not support speed selection for lines equipped with Dual Speed Modems. It also does not support Switched Network Backup.

### ***Remotely Attached Terminals***

VTAM can control remotely attached terminals only if they are attached via a communications controller in network control mode. Start-stop and BSC terminals can be attached either to a local communications controller or to a remote communications controller. Logical units can only be attached via a local communications controller.

Appendix B lists all devices and features that can be used as remote terminals by VTAM.

Because VTAM uses the NCP to control and communicate with remotely attached terminals, most requirements, restrictions, or limitations pertaining to device features are those of the NCP, not of VTAM. (See the *NCP Generation* publication for details on device requirements, restrictions, and limitations for NCP.)

## Operating System Requirements

This section discusses VTAM's requirements for each operating system (DOS/VS, OS/VS1, and OS/VS2) in which VTAM can be included.

### *Upward Compatibility*

To facilitate teleprocessing growth, VTAM is upward compatible from DOS/VS to OS/VS1 to OS/VS2. That is, the application-program macro language is upward compatible. In addition, procedures for defining the telecommunication network and tailoring VTAM are similar for all three systems. So too, VTAM's network-operator commands are similar across the three systems.

Although VTAM does provide upward compatibility for telecommunications, it uses the facilities of the operating system under which it is executing. VTAM's requirements for each of these systems are detailed in the remainder of this chapter.

### *Requirements for DOS/VS*

Generating VTAM in a DOS/VS system requires that VTAM be specified as a teleprocessing access method in the SUPVR macro instruction for DOS/VS system generation. VTAM also requires the inclusion of some DOS/VS facilities that would be optional if VTAM were not in the system. Of these facilities required by VTAM, multitasking support must be specified by the installation during system generation. The other options are generated automatically if VTAM is specified; they include:

- Support for the use of the STXIT macro instructions (all options) by problem programs.
- Storage-management support for the GETVIS and FREEVIS macro instructions.
- Use of the PFIX and PFREE macro instructions for fixing and freeing pages.
- Inclusion of the relocating loader.
- Support for the time-of-day clock.

All locally attached devices in the VTAM network must be identified during the generation of the operating system or added at IPL (initial program loading) time. These devices are the locally attached 3270s and the locally attached communications controllers used by VTAM. Telecommunication lines and remotely attached devices need not be defined at system generation if they are to be used only through VTAM. If they are to be used directly by other access methods, they must be defined according to the requirements of those access methods; remotely attached devices so defined are still available to VTAM through the NCP. Remotely attached devices are defined to VTAM as explained in "VTAM Definition," in Chapter 3.

A DOS/VS system with VTAM must have at least two partitions: one for VTAM and one for VTAM application programs. Additional partitions may be needed for other VTAM application programs or for programs unrelated to VTAM. The partition containing VTAM must have a priority higher than any other partition that contains a VTAM application program. VTAM also requires three DOS/VS tasks in addition to any tasks required for application programs.

VTAM is started under DOS/VS by entering an EXEC command or job control statement for the partition in which VTAM is to run. In addition to the EXEC statement, an ASSIGN command or statement is required if a local communications controller is to be included (that is, if remote terminals are to be included) in the VTAM network. The assignment must make logical unit SYS000 unassigned for the duration of the VTAM job step. Any other commands or job control statements required to start VTAM depend upon factors such as system generation and telecommunication options to be used. See "VTAM Data Sets," later in this chapter, for library and file requirements for VTAM under DOS/VS.

## **Requirements for OS/VS**

To generate VTAM in an OS/VS system requires that VTAM be specified as a teleprocessing access method in the DATAMGT macro instruction for OS/VS system generation.

All locally attached devices in the VTAM network must be identified at system generation. These devices are the locally attached 3270s and the locally attached communications controllers used by VTAM. Telecommunication lines and remotely attached devices need not be defined at system generation if they are to be used only through VTAM. If they are to be used directly by other access methods, they need to be defined according to the requirements of those access methods; remotely attached devices so defined are still available to VTAM through the NCP. Data set requirements for OS/VS are discussed in "VTAM Data Sets," later in this chapter. Remotely attached devices are defined to VTAM as explained in "VTAM Definition," in Chapter 3.

A cataloged procedure must be created for VTAM. This procedure must contain DD statements for SYS1.VTAMLST and SYS1.VTAMLIB. In addition, the procedure must contain statements for any NCP data sets that are to be used by VTAM. It must also contain a DD statement for SYS1.VTAMOBJ, if this data set is to be used, and DD statements for the TOLTEP data sets, if TOLTEP is to be invoked. Locally attached 3270s, communications controllers, and remotely attached terminals and devices that are part of the VTAM telecommunication network do *not* require DD statements.

Multiple console support (MCS) can be used with VTAM. Any MCS console that is to be used by the VTAM network operator must be assigned a command authority of two and a routing code of eight.

It is not recommended that VTAM be run with the Dynamic Support System (DSS). The affects of DSS on VTAM includes DSS time delays that may result in NCPs in the VTAM network automatically closing down the network. Refer to the publication *OS/VS Dynamic Support System*, GC28-0640, for details on DSS.

## **Network Control Program Requirements**

This section describes the requirements placed upon an NCP by VTAM. It describes the relationship between NCP generation and VTAM definition. It also discusses VTAM's support of switched networks and VTAM's impact on the NCP with PEP.

### ***Introduction to the Network Control Program***

VTAM uses the network control mode of the network control program/VS (NCP) in the IBM 3704 and 3705 Communications Controllers to control and communicate with remotely attached devices. The NCP is generated with NCP generation macro instructions. These same instructions are then used as definition statements to define the remote network to VTAM. See "VTAM Definition," in Chapter 3, for more information on using the same deck of statements both to generate an NCP and to define the NCP and its network to VTAM.

The remainder of this chapter contains details on VTAM's use of the NCP. Refer to the publication *Introduction to the IBM 3704 and 3705 Communications Controller*, GA27-3051, for a description of the NCP and its facilities. Refer to the *NCP Generation* publication for details on NCP generation macro instructions and NCP generation procedures.

### ***NCP Functions Required by VTAM***

Several types of NCPs may be needed in a VTAM telecommunication system:

- An NCP for a local communications controller to which no remote communications controller is attached.

- An NCP for a local communications controller to which a remote communications controller is attached.
- An NCP for a remote communications controller.
- An NCP with the partitioned emulation programming (PEP) extension.

Whatever the NCP or NCPs used, some functions that are optional for the NCP are required by VTAM. These functions are part of the NCP's dynamic control facilities which are created during NCP generation, and they are specified in the NCP's SYSCNTRL statement. The facilities that are required by VTAM enable the access method to dynamically request the NCP to perform any of the following activities:

- Activate or deactivate groups, lines, or devices.
- Provide the status of a device.
- Alter the sequence of NCP commands for a particular device.
- Change the block processing routines associated with a particular device.

(Needed only for start-stop or BSC lines).

- Change mode settings for a device.
- Disconnect a dial connection.
- Request a device to yield control of a line.
- Reset a device if data transfer has not taken place.
- Reset a device if data transfer has taken place.
- Reset a device immediately.

### *Defining the NCP*

Defining an NCP and its remote network to VTAM includes specifying the following information in the definition (also the generation) statements:

- The capabilities of the NCP.
- The interface between the NCP and VTAM.
- The network configuration.

This information is supplied to VTAM via NCP generation macro instructions plus three additional VTAM definition statements; PCCU, VIDLIST, and VTERM. These three statements are defined below.

#### *PCCU Statement*

The PCCU statement is used only by VTAM. This statement identifies the communications controller into which the NCP would normally be loaded when activated. One PCCU statement is required for each NCP defined to VTAM.

This statement also defines the functions VTAM is to provide for the NCP. These functions can include:

- Dumping the communications controller automatically following an unrecoverable error in that controller. If a dump is to be taken, a pointer to the name of the data set to contain that dump is specified in the PCCU statement.
- Reloading and restarting the NCP automatically following an unrecoverable error in the communications controller.
- Invoking the NCP initial test routine automatically prior to loading the NCP. (Initial testing is optional only for NCPs for locally attached communications controllers; initial testing is performed automatically whenever a remotely attached communications controller is loaded.)



If the NCP is for a remotely attached communications controller, this statement also provides a link to the NCP for the locally attached communications controller. A PCCU statement for an NCP for a remote controller, specifies the name of a INNODE statement in another NCP deck. This deck defines the NCP for the local communications controller, and the INNODE statement defines the remote controller.

#### *VIDLIST statement*

The VIDLIST statement is used only by VTAM. It defines identification sequences for BSC terminals and for TWX terminals. These are the identifications that are to be verified by VTAM, as opposed to those to be verified by the NCP and specified in the IDLIST statement. The VIDLIST statement specifies valid identifications along with the name of the terminal. The terminal name is the name of a TERMINAL statement that defines the terminal that can enter that identification.

The VIDLIST statement indicates those sequences to be verified by VTAM. It is meant to be used in conjunction with the NCP's IDLIST statement to provide identification verification. See "Telecommunication Security," later in this chapter, for details on using the VIDLIST statement.

#### *VTERM Statement*

The VTERM statement is used only by VTAM. It provides an identity and logon information for specific terminal types that are using a multiple-terminal-access (MTA) line. (The MTA facility of the NCP enables a variety of dissimilar start-stop terminals to use the same network of switched lines.)

The VTERM statement enables an installation to specify a name for a single type of terminal. It can also be used to specify the name of an application program to receive an automatic logon request whenever a terminal using this name is available. The statement can also be used to specify an interpret table for any terminal using this name. (See "VTAM Definition," in Chapter 3, for information on automatic logons and on interpret tables.)

The VIDLIST and the VTERM statements are optional statements for VTAM definition; the PCCU statement is required.

In addition to using the PCCU, VIDLIST, and VTERM definition statements, VTAM requires information contained in the NCP generation macro instructions (also called VTAM definition statements). Most information required by VTAM is also required by the NCP, although some additional data is required by VTAM alone. (Refer to the *NCP Generation* publication for details on the GROUP, LINE, CLUSTER, TERMINAL, and COMP statements as they apply to start-stop and BSC lines and terminals.)

VTAM uses the NCP's teleprocessing network configuration macro instructions plus VTERM statements to define the start-stop and BSC remote-device configuration. Using the GROUP, LINE, CLUSTER, TERMINAL, COMP, and VTERM statements, an installation can specify to VTAM the following items for start-stop and BSC terminals:

- Automatic logon and interpret table requirements.
- A description of the features for a remotely attached 3270 terminal.
- The initial status of a terminal or a cluster control unit when the NCP is activated by VTAM.
- Whether, after a successful retry operation for a terminal, the message block is to be accepted.
- The buffer limit for a terminal. (See "VTAM Buffering," in Chapter 3, for a description of how buffer limits are established.)

- The name of each terminal. The name of the terminal is usually the name of the `TERMINAL` or `COMP` statement. If the `TERMINAL` statement defines a *logical connection terminal* (that is, if it contains the `CTERM=YES` operand), the name of the terminal must be specified via an additional operand, `UTERM`. The name specified by `UTERM` is used only by VTAM and applies to terminals on a switched line. See “Support for Switched Networks,” below, for an explanation of the use of the `UTERM` name. Refer to the *NCP Generation* publication for an explanation of the `CTERM` operand.
- The name of each group, line, and cluster control unit (if any). (Names are specified in the `GROUP`, `LINE`, and `CLUSTER` statements, respectively.)

Using the `GROUP`, `LINE`, `CLUSTER`, and `LU` statements, an installation can define teleprocessing subsystems, which are attached via SDLC lines, to VTAM. The `LU` statement (which, for logical units, is the functional equivalent of the `TERMINAL` statement) describes logical units to VTAM and the NCP. In the case of teleprocessing subsystems, the `GROUP` statement defines a group of SDLC lines; the `LINE` statement defines a single SDLC line; the `CLUSTER` statement defines a cluster control unit for a teleprocessing subsystem (for example, a 3601 for a 3600 Finance Communication System); and the `LU` statement defines a logical unit (refer to Figure 2-4 for the relationship between VTAM logical units and teleprocessing subsystems).

Using these statements, the installation can specify to VTAM the following items for logical units:

- Automatic logon and interpret table requirements.
- The initial status of the logical unit and cluster control unit when the NCP is activated by VTAM.
- The buffer limit for the logical unit. See “VTAM Buffering,” in Chapter 3, for a description of how buffer limits are established.
- Pacing requirements for each logical unit.
- The name of each group, line, cluster control unit, and logical unit. (Names are specified in `GROUP`, `LINE`, `CLUSTER`, and `LU` statements, respectively.)

Logical units are initially activated by VTAM only if they are defined as initially active and their SDLC cluster controllers are initially active. Cluster control units, in turn, are initially activated only if they are defined as initially active and if they have been manually loaded prior to activation by VTAM.

VTAM enables the installation to control the rate of data flow through the network path joining a VTAM application program and a logical unit. This control is called pacing. Pacing is intended to protect a node that is receiving data from being overloaded by too much input. Using the specifications supplied in an `LU` Statement by the installation, both VTAM and the NCP can limit the number of messages transmitted to a particular logical unit before a response is returned that the messages have been received. For each logical unit, the installation can specify pacing requirements for the transmission of messages from VTAM to the NCP and for the transmission of messages from the NCP to the logical unit.

### ***Support for Switched Networks***

This section describes VTAM’s support for call-in, call-out, and call-in/call-out terminals. (Switched-network support is provided only for start-stop and BSC terminals as specified in Appendix B.) Also included are discussions on network-operator considerations that apply to controlling switched networks. This section is meant to augment the discussion on NCP support for switched networks provided in the *NCP Generation* publication.

## Call-in Terminals

Call-in terminals have VTAM-definition requirements that affect some network-operator and application program activities. Understanding these effects requires an understanding of VTAM's support of call-in terminals.

VTAM uses the concept of a port to support call-in terminals. As noted in the *NCP Generation* publication, each call-in line must have a `TERMINAL` statement with a `CTERM=YES` operand. These statements represent logical connections to the NCP, but they represent ports to VTAM.

For VTAM to accept a call-in request over a switched line, the port for that line (in addition to the other nodes in the path) must be active. Ports are activated automatically when an NCP is activated and loaded by VTAM. Thereafter, ports can be activated or deactivated using the network-operator's `VARY` command.

The name of a port is the name of the `TERMINAL` statement with the `CTERM=YES` specification. This name is used by the network operator to address the port, but it is not used by an application program because application programs connect only to terminals.

Note that for a switched line, a port is a minor node: It is defined to VTAM and can be addressed. In addition, the port must be active (as must other path elements such as lines and cluster control units) for a terminal to be connected with an application program.

The `TERMINAL` statement that defines a port may also contain a definition of a terminal. Such a statement is used as a definition of a terminal only if the terminal calling in can not be identified and associated with another `TERMINAL` or `VTERM` statement. (This identification would be provided via either the NCP's MTA facility or via VTAM's and the NCP's BSC, and TWX, identification facilities.) That is, if a terminal calling in over a switched line cannot be identified, VTAM attempts to apply the terminal description on the port (`TERMINAL`) statement for that line to the terminal. Thus, information such as terminal-type, automatic-logon specifications, and initial status is applied to the terminal calling in.

For the description on the port statement to be applied to the terminal, a `UTERM` operand must be specified on that statement. The name in the `UTERM` specification is the name of the terminal (any unidentified terminal) calling in over the line serviced by the port. See "Defining the NCP," earlier in this chapter, for a description of the `UTERM` specification on the `TERMINAL` statement.

An application program wishing to be connected to any unidentified terminal calling in over a specific line uses, as the name of the terminal, the `UTERM` name, not the name of the `TERMINAL` statement itself. Likewise, an automatic-logon specification on a port statement is applied to any unidentified terminal calling in over that line; the terminal is logged on to the program specified (be it the network solicitor or an application program).

In summary, the following should be considered when planning to use VTAM's support for call-in terminals:

- For MTA lines, the `VTERM` statement can be used to provide identification and logon information for each type of terminal. If a `VTERM` statement is not used, an MTA terminal is defined by the port statement (if that statement has a `UTERM` specification).
- For BSC (and TWX) terminals, the `IDLIST` and the `VIDLIST` statements can be used to distribute identification responsibility between the NCP and VTAM.
- For port definitions, the `TERMINAL` statement with the `CTERM=YES` parameter defines a port.

- For unidentified terminals, a UTERM name must be specified on a port statement if VTAM is to connect unidentified terminals calling in over the line serviced by the port.

#### **Call-out Terminals**

VTAM has no special requirements for supporting call-out terminals. Using VTAM-definition and NCP facilities, an installation can specify that a terminal is to be dialed automatically or manually by the network operator. The dial digits must be specified at NCP generation. For automatic dialing, the numbers are dialed by the NCP. For manual dialing, VTAM transmits a message (containing the dialing instructions) to the network operator.

#### **Call-in/Call-out Terminals**

Planning for terminals that are capable of both calling in and calling out requires special consideration. Such terminals might be represented twice to the NCP and to VTAM.

If a terminal can be identified during a call-in operation through BSC (or TWX) identification facilities, it is defined for both call-in and call-out operations by the same TERMINAL statement.

If a call-in terminal is unidentified or is an MTA terminal, it is defined by either the UTERM name in a TERMINAL statement or by a VTERM statement (MTA terminals only). If the same terminal can be called, it must have another TERMINAL statement defining its call-out characteristics. Thus, the terminal has two definitions and two names: one for calling in, the other for calling out.

Although the call-out definition applies to a specific terminal, the call-in definition can apply to any valid, but unidentified (including MTA) terminal calling in.

An activation or deactivation request received from the network operator and directed to one of the terminal definitions does not affect the other definition. For example, a deactivation request specifying the call-out name does not affect the terminal's ability to call in. If such a request were issued, the terminal could still be used to call in even though a call-out operation would be prohibited by VTAM.

Similar considerations apply for application programs connecting with terminals capable of both being called and calling in. If an application program were to connect with a terminal (MTA or unidentified) that had called in, it would connect using the call-in name. Then, if the application program were to disconnect the terminal and subsequently attempt to reconnect it by dialing out, the name of the terminal specified in call-out definition would have to be used in the connection request.

#### ***Partitioned Emulation Programming (PEP) Considerations***

VTAM can use the NCP with the partitioned emulation programming (PEP) extension. When using the NCP with PEP, VTAM is generally unaware of the emulation function in the communications controller; that is, VTAM does not provide services for the network serviced by emulation mode.

VTAM does provide three facilities that affect the emulation mode of an NCP with PEP: dumping the communications controller, reloading the controller, tracing emulation lines, and changing line assignments.

If the network operator invokes the NCP dump utility program (using VTAM's MODIFY command), the dump disrupts the emulation function as well as the network control function when an NCP with PEP resides in the communications controller. After a dump is taken, the NCP has to be reloaded before it can be used. (An NCP with PEP is loaded automatically by VTAM when the access method activates an NCP major node defining a network control function that is part of an NCP with the PEP extension.)

VTAM also provides support for the restart capability of the NCP. If an error is encountered in a communications controller and this error causes VTAM to reload the controller, the reloading disrupts emulation because the entire NCP is reloaded, including the emulation function.

If VTAM activates an NCP with PEP, the access method's MODIFY command can be used to activate NCP line traces for lines that are operating under emulation mode as well as for those that are operating under network control mode. The trace records for emulation lines are handled like those for network control lines. Records are written to the trace data set used by VTAM and are printed using the VTAM trace-print utility program for DOS/VS or the operating systems' print-dump service aids for OS/VS.

VTAM also affects the emulation function in an NCP with PEP by changing line assignments. Using PEP, lines can be assigned to network control mode, emulation mode, or to both. If a line can be assigned to both, VTAM manages this assignment. VTAM's support for such lines is as follows:

- If the line is not active for VTAM, it is automatically assigned to emulation mode.
- Line assignments are changed in response to activation and deactivation requests from the network operator for VTAM. Lines are assigned to network control mode when they are activated by VTAM, and they are reassigned to emulation mode when they are deactivated by VTAM.
- A request to VTAM to activate a line is rejected if that line is currently being used by another access method through emulation mode. The line assignment is changed only when the line is no longer being used in emulation mode. The line is no longer in use through emulation mode when the job step to which it is allocated ends (DOS/VS and OS/VS1) or when the job step issues a CLOSE macro instruction for the line (OS/VS2, with dynamic allocation).

## VTAM Data Sets

VTAM uses a number of data sets (files in DOS/VS) to support a telecommunication system. These data sets contain such items as VTAM modules, VTAM definition statements, NCP modules, and RAS records. Data set requirements vary depending upon the operating system in which VTAM is being used.

### *Data Sets for VTAM Under OS/VS*

In an OS/VS system, VTAM uses NCP data sets, operating system data sets, and data sets devoted entirely to VTAM itself. Figure 7-1 is a table of all the data sets used by VTAM under an OS/VS system. This table is divided into columns; each defined as follows:

- Name—specifies the name assigned to the data set.
- VTAM Contents—indicates what information in the data set is used by VTAM.
- When Created—specifies the latest time that the data set can be created.
- JCL—specifies whether a DD statement must be provided for the data set in the VTAM start procedure.
- Comments—provides additional information relevant to the data set. If the data set is specified as “required,” it is required by VTAM. If it is specified as “optional,” it is not required by VTAM.

**Note:** If an optional data set is not included in a VTAM system, the contents of that data set, and therefore the associated function, are not available to VTAM.

Refer to the publications *OS/VS1 Planning and Use Guide*, GC24-5090, or *OS/VS2 Planning Guide for Release 2*, GC28-0667, for more information on the operating system

Name	VTAM Contents	When Created	JCL	Comments
<i>VTAM Data sets</i>				
SYS1.VTAMLIB	Load modules for VTAM; see note 1	System generation	Yes	Only the VTAM load modules are required
SYS1.VTAMLST	VTAM definition statements and start options	Prior to starting VTAM	Yes	Required. See note 2
SYS1.VTAMOBJ	VTAM network configuration table	When VTAM is started	Yes	Optional. See note 3
<i>Operating System Data sets</i>				
SYS1.DUMP	Records for SVC dumps	IPL	No	Optional
SYS1.LINKLIB	VTAM initialization module, used when VTAM is started	System generation	No	Required
SYS1.LOGREC	VTAM error records	System generation	No	Required
SYS1.LPALIB	VTAM load modules to be loaded into the shared link-pack area	System generation	No	Required. An OS/VS2 data set only
SYS1.MACLIB	VTAM macro definitions	System generation	No	Required
SYS1.NUCLEUS	VTAM resident SVCs and abnormal termination modules	System generation	No	Required
SYS1.PARMLIB	VTAM space parameter for IPL	System generation	No	Required. An OS/VS1 data set only
SYS1.PROCLIB	VTAM cataloged procedure	System generation	No	Required
SYS1.SVCLIB	VTAM non-resident SVCs and ERPs for locally attached devices	System generation	No	Required. An OS/VS1 data set only
SYS1.TRACE	GTF trace records for VTAM	System generation	No	Optional

**Notes:**

1. SYS1.VTAMLIB is referred to as the *VTAM load module library*. For OS/VS1, SYS1.VTAMLIB contains VTAM load modules for the VTAM virtual partition; for OS/VS2, it contains VTAM load modules for the VTAM private address space. In both OS/VS1 and OS/VS2, this library can contain the interpret table, or tables, containing logon descriptions and any installation-coded logon-routines specified in these tables; authorization and accounting exit-routines; and the network solicitor.
2. SYS1.VTAMLST is referred to as the *VTAM definition library*. Definition statements for each major node are filed in this library. Each major node is a separate member of SYS1.VTAMLST.  
Start options are filed in members under the name of either ATCSTRxx or ATCCONxx (where xx are two-digit numbers specified by the installation). ATCSTRxx members contain lists of major nodes to be activated automatically when VTAM is started.  
See "VTAM Definition", in Chapter 3, for details on VTAM definition statements and VTAM start options.
3. When a major node is activated, VTAM builds a table describing the node from the information supplied by definition statements. When a major node is deactivated, its table is deleted by VTAM. These tables are used to maintain the current status of all minor nodes in the telecommunication system.  
To reduce the time needed to construct these tables, VTAM stores a copy of each table on SYS1.VTAMOBJ the first time each major node is activated. This copy is then used whenever the major node is again activated.  
*Note:* A major node can be modified merely by modifying its definition statements and refileing them under the same member name on SYS1.VTAMLST. If a member is refiled, the copy of the corresponding table on SYS1.VTAMOBJ must be deleted (using an operating-systems utility program that can delete a member of a BPAM data set). If the copy is deleted, the next time the major node is activated, VTAM builds a new table (based on the modified definition) and stores a new copy on SYS1.VTAMOBJ.
4. These data sets are referred to as the *NCP load module libraries*.

Figure 7-1 (Part 1 of 2). Table of Data Sets Used by VTAM Under OS/VS

Name	VTAM Contents	When Created	JCL	Comments
<i>NCP Data sets</i>				
NCP dump	Dump records for the NCP	VTAM start time	Yes	Required if VTAM is requested to provide a dump of an NCP. One such data set must be provided for each NCP that is to be dumped simultaneously.
NCP load library	NCP load modules	NCP generation	Yes	One such data set must be created for each NCP used by VTAM. See note 4
INITEST	Local communications controller pre-IPL testing modules	NCP generation	Yes	Required if the controller is to be tested automatically prior to the loading of the NCP by VTAM
<i>TOLTEP Data sets</i>				
TOLTEP CDS	Configuration data sets	Prior to invoking TOLTEP	Yes	Required if TOLTEP is to be executed
TOLTEP OLTs	Online terminal tests for TOLTEP	Prior to invoking TOLTEP	Yes	Required if TOLTEP is to be executed

**Notes:**

1. SYS1.VTAMLIB is referred to as the *VTAM load module library*. For OS/VS1, SYS1.VTAMLIB contains VTAM load modules for the VTAM virtual partition; for OS/VS2, it contains VTAM load modules for the VTAM private address space. In both OS/VS1 and OS/VS2, this library can contain the interpret table, or tables, containing logon descriptions and any installation-coded logon-routines specified in these tables; authorization and accounting exit-routines; and the network solicitor.
2. SYS1.VTAMLST is referred to as the *VTAM definition library*. Definition statements for each major node are filed in this library. Each major node is a separate member of SYS1.VTAMLST.  
Start options are filed in members under the name of either ATCSTRxx or ATCCONxx (where xx are two-digit numbers specified by the installation). ATCSTRxx members contain lists of major nodes to be activated automatically when VTAM is started.  
See "VTAM Definition", in Chapter 3, for details on VTAM definition statements and VTAM start options.
3. When a major node is activated, VTAM builds a table describing the node from the information supplied by definition statements. When a major node is deactivated, its table is deleted by VTAM. These tables are used to maintain the current status of all minor nodes in the telecommunication system.  
To reduce the time needed to construct these tables, VTAM stores a copy of each table on SYS1.VTAMOBJ the first time each major node is activated. This copy is then used whenever the major node is again activated.  
*Note:* A major node can be modified merely by modifying its definition statements and refiling them under the same member name on SYS1.VTAMLST. If a member is refiled, the copy of the corresponding table on SYS1.VTAMOBJ must be deleted (using an operating-systems utility program that can delete a member of a BPAM data set). If the copy is deleted, the next time the major node is activated, VTAM builds a new table (based on the modified definition) and stores a new copy on SYS1.VTAMOBJ.
4. These data sets are referred to as the *NCP load module libraries*.

Figure 7-1 (Part 2 of 2). Table of Data Sets Used by VTAM Under OS/VS

data sets listed in Figure 7-1. Refer to the publication *Introduction to the IBM 3704 and 3705 Communications Controllers*, GA27-3051, for additional information on NCP data sets listed in Figure 7-1.

**Data Requirements for VTAM Under DOS/VS**

A file in DOS/VS is comparable to a data set in OS/VS. VTAM has file requirements in a DOS/VS system similar to the data set requirements in OS/VS. Figure 7-2 is a table that shows the file requirements for VTAM under DOS/VS. This table is divided into columns; each column is defined as follows:

- File Name—specifies the name of the file. Except as indicated in Figure 7-2, this name must be specified as shown.
- File ID—specifies the identification of the file on the volume.
- Contents—indicates what in this file is used by VTAM.
- Comments—provides additional information relevant to the file. If the file is specified as “required” it is required by VTAM.

Refer to the publication *Introduction to DOS/VS*, GC33-5370, for additional information on the DOS/VS logical units and files. Refer to the publication *Introduction to the IBM 3704 and 3705 Communications Controllers*, GA27-3051, for additional information for NCP data requirements.

VTAM also has data requirements pertaining to DOS/VS libraries. Whether these libraries are system or private libraries depends upon system generation options selected by the installation. Library requirements for VTAM under DOS/VS are described in Figure 7-3.

File Name	File Identification	Content	Comments
DIAGFILE	INITST	Communications controller pre-IPL testing modules	Required if the controller is to be tested automatically prior to the loading of the NCP by VTAM
NCP load file	Note 1	NCP load modules	One such file must be created for each NCP used by VTAM. See note 2
NCPDUMP	Note 3	Dump records for the NCP	Required if VTAM is requested to provide a dump of an NCP. One such file must be provided for each NCP that is to be dumped simultaneously
TRFILE	TRFILE	VTAM trace records	Required. Must be assigned to SYS001

**Notes:**

1. File names and file identifications are created during NCP generation.
2. These files are referred to as the *NCP load libraries* in this publication.
3. File identifications are created during NCP generation.

Figure 7-2. Table of Files Used by VTAM under DOS/VS



DOS/VS Library	Content	Comments
Core image library	VTAM modules; NCP tables and block handler routines; configuration data sets and online terminal tests for TOLTEP	Required. See Note 1
Procedure library	VTAM start procedure	Optional
Source statement library	VTAM definition statements and start options	Required. See note 2

*Notes:*

1. When this publication refers to the *VTAM load module library*, it is referring to the VTAM contents of this library. As the VTAM load module library, this library also contains the interpret table, or tables, containing logon descriptions and any installation-coded logon-routines specified in these tables; authorization and accounting exit-routines; and the network solicitor.
2. When this publication refers to the *VTAM definition library*, it is referring to the VTAM contents of this library. Definition statements for each major node are filed in this library. Each major node is a separate book of the source statement library.

Figure 7-3. VTAM's Use of Libraries under DOS/VS

## Sharing Telecommunication Resources

One way to use a telecommunication system effectively is to keep at a minimum the amount of time a user of the system must wait for a system resource. Sharing resources among several users can reduce the amount of wait time and permit more efficient use of these resources.

VTAM permits resources in its telecommunication system to be shared. The degree of sharing depends upon such factors as the resources themselves and the installation's management of these resources.

### *Resources that can be Shared*

Using VTAM facilities, application programs can share terminal components, terminals, cluster control units, communications controllers, NCP facilities, telecommunication lines, and data channels. Of these elements, terminals (or components) and application programs can be thought of as end nodes, while the remainder of the elements are part of the path connecting the end nodes.

Sharing of end nodes is different from the sharing of path elements. End nodes are shared per connection; path elements can be shared simultaneously. That is, an active terminal (an end node) can be connected to or queued for logon to only one application program at a time. If an active terminal is not connected or queued for logon to an application program, it is available for connection to any application program.

On the other hand, VTAM does not explicitly connect path elements to end nodes. Instead, VTAM employs path elements on behalf of end nodes only as long as needed to complete a specific I/O request. For example, more than one terminal can be attached to the same multipoint line, and each terminal can be connected to, or queued for logon to, a different application program. Thus, the line (path element) is being used simultaneously by several applications.

### *Sharing and Managing Resources*

While VTAM allows many resources to be shared, the installation can restrict some sharing for such reason as performance or security. For example, a high-priority terminal might be attached to a nonswitched, point-to-point, line to improve access to it, or

### Managing Resources Through VTAM Definition

connection to a security-sensitive application program might be limited to only certain terminals.

The following list specifies the areas in which an installation can affect resource sharing:

- Defining the network.
- Generating the NCP.
- Executing application programs.

Resource management as it pertains to each of these areas is discussed below.

The telecommunication network that VTAM manipulates is the one presented to it through network definition. VTAM provides a number of facilities in network definition that enable an installation to affect the sharing of telecommunication resources. These facilities include:

- Logon (automatic and terminal-initiated)
- Application program authorization.
- Authorization exit-routine.
- Initial status of nodes.

The effects of each facility on sharing are discussed below. Details on these facilities are provided in "VTAM Definition," in Chapter 3.

**Sharing Through Logon:** Using the terminal-initiated logon capability of VTAM, an installation can make all input/output terminals available to all application programs. It would be possible to define a network in which application programs could connect with input/output terminals only in response to a terminal-initiated logon. (Input-only and output-only terminals are available for connection via other means, such as automatic logon or acquisition).

In practice, some of the terminals might not be available to all application programs. For reasons such as security, some of the terminals might be available only to a restricted set of application programs.

Restricting the availability of a terminal can be accomplished by:

- Limiting the number of valid logon messages for a terminal that is to use the terminal-initiated logon capability of VTAM.
- Making the terminal available for connection only through the SIMLOGON macro instruction or the ACQUIRE option of the OPNDST macro instruction, and then controlling which application programs can issue those macro instructions.
- Prohibiting some connections through the installation-coded authorization exit-routine.

**Application-Program Authorization:** Some facilities in VTAM require prior authorization before they can be used by an application program. These facilities can reduce the availability of resources and should therefore be used judiciously:

- **Block processing**—As compared to message or transmission processing, block processing can result in a greater number of I/O interruptions for the application program. More interruptions, in turn, can result in more delays for path elements such as telecommunication lines and data channels. These delays then decrease the availability of these resources to other application programs.

- **Passing connections**—Passing a terminal to another application program queues the terminal to that application program. As long as the terminal is on the queue, it is not available for communication, either to that application program or to any other application program. Care should be taken to ensure that terminals are on the connection queue no longer than necessary.

**Authorization Exit-Routine:** Although VTAM provides the potential for any terminal to be connected to any application program, an installation can code an authorization exit-routine restricting specific connections. VTAM passes control to this exit-routine during the processing of connection and disconnection requests.

**Initial Status of Nodes:** Using VTAM definition statements, an installation specifies the initial status of each terminal and cluster minor node in the VTAM system. These minor nodes are defined as initially active or initially inactive. If a minor node is defined as initially inactive, the network operator must explicitly activate it (via VTAM's VARY command) before it can be used by VTAM. On the other hand, if a minor node is defined as initially active, network-operator intervention is not required, and the node may be made available sooner.

Whatever the initial status specified for a minor node, the minor node cannot be activated until its major node is active. Like minor nodes, major nodes can be activated explicitly by the network operator, or they can be activated automatically (at VTAM start time) without the need of network operator intervention.

### Managing Resources Through NCP Generation

Because VTAM uses the NCP to communicate with remotely attached devices, options generated in the NCP can impact the sharing capabilities offered by VTAM. For example, if an NCP session limit specified for a multipoint line is not adequate to service all devices on that line simultaneously, some devices may be effectively unavailable for communications.

Refer to the *NCP Generation* manual for more information on NCP options.

### Managing Resources Through Application Programs

Once a VTAM system has been generated and defined, the availability of resources can be influenced by the application programs' use of VTAM. Although Chapter 5 discusses the functions that can be performed in an application program using VTAM, this section summarizes some of the ways by which an application program can affect the degree of sharing.

As noted earlier in "Managing Resources Through Network Definition," an application program using block processing can limit the availability of some path elements. In addition, an application program can monopolize telecommunication resources that normally would be shared:

- As long as a terminal is connected to or queued for connection to an application program, it is unavailable to other application programs.
- As long as an application program and a terminal are connected, the NCP sessions available for a multipoint line (in the case of remotely attached terminals) is reduced by one.

Connection and NCP session, as they pertain to sharing resources, are discussed below.

**Connection and Terminal Sharing:** To achieve maximum use of terminals, the installation can instruct application programmers to ensure that terminals are connected, or queued for logon, only as long as necessary. Using VTAM's OPNDST and CLSDST macro instruction, an application program can connect with a terminal only when an actual

input or output operation is needed. This feature is particularly useful for application programs with very long delays between message transmissions. If a program handling such applications could tolerate non-continuous connections, terminals could be released during the delays for use by other application programs.

**NCP Session and Line Sharing:** While an application program and a terminal are connected in basic-mode, it is possible for them to monopolize the telecommunication line. The first I/O request from an application program for a terminal connected in basic-mode initiates an NCP session for the terminal. This session is continued until the application disconnects the terminal. Thus, for most application programs, the duration of an NCP session nearly equals the duration of a connection.

The relationship between connection and the NCP session is particularly significant if the line is multipoint and the session limit (as specified in NCP generation) is less than the number of terminals on that line or if the system is DOS/VS and the line is point-to-point with attached components.

In the case of multipoint lines, it is possible for the number of simultaneous connections to equal the specified NCP-session limit. If they are equal and the session limit is less than the number of attached terminals, communication with a terminal not currently in session is prohibited until a current session on the line is ended by the disconnection of another terminal. If the session limit for a multipoint line equals the number of terminals on that line, contention for the NCP session is reduced.

In the case of components on a point-to-point line (under DOS/VS), only one session is permitted at a time for that line.

## Telecommunication Security Through VTAM

This section discusses how to establish and maintain telecommunication security in a VTAM system. It identifies and describes the facilities in VTAM that help an installation to control access to privileged or sensitive data and resources.

### *Introducing Telecommunications Security*

Using VTAM facilities, an installation can identify and control sensitive resources. Essentially, an installation can identify and control the use of specified terminals and application programs.

Identification of resources to be controlled is performed during VTAM definition; control of the access to these resources can be performed at various points in the telecommunication network and during various stages of telecommunication processing. The installation specifies at what point or points this control is to be exercised.

Depending upon the desired type of control, an installation can specify that control of sensitive resources be performed at one or more of the following points:

- In the communications controller network control program (NCP).
- In VTAM.
- In an installation-coded authorization exit-routine.
- In application programs.

The ways in which a VTAM installation can control sensitive resources include:

- Verifying terminal identifications. (binary synchronous communications and TWX terminals only)

- Verifying logon requests.
- Controlling the connections between selected application programs and terminals.
- Controlling which application programs can use VTAM.
- Restricting an application program's use of selected VTAM facilities.
- Protecting sensitive data being transmitted by VTAM.

The remainder of this section discusses each control facility in detail.

### ***Identification Verification***

Using the IDLIST (OS/VS only) and the VIDLIST definition statements, an installation can identify binary synchronous communications (BSC) or TWX terminals attached to switched lines. Both definition statements are placed in the generation (and definition) deck for the communications controller network control program (NCP) for virtual systems. The IDLIST definition statement indicates those identifications to be verified by the NCP. (The IDLIST statement is not valid for an NCP that is to use VTAM under DOS/VS.) The VIDLIST statement indicates those identifications to be verified by VTAM in the host computer. (See "Network Control Program Requirements," earlier in this chapter, for more information on these definition statements.)

In VTAM under OS/VS, an installation can distribute verification authority between an NCP in a communications controller and VTAM in the host computer. Thus, to conserve NCP resources, an installation might specify in an IDLIST statement only those terminal identifications that are used heavily, leaving for VTAM the verification of less frequently used terminals. In VTAM under DOS/VS, all verification must be done by VTAM in the host computer.

Using IDLIST definition statements, an installation can specify the following information for the NCP:

- The identification character string for each BSC or TWX terminal on a switched line to be verified by the NCP.
- The symbolic name to be assigned to the terminal entering a verified identification character string. A unique name can be specified for each verifiable character string.
- The action to be taken by the NCP upon encountering a character string that is not described in an IDLIST definition statement. The NCP can either pass the unverified identification to VTAM for further verification processing or stop communicating with the terminal.

Using the VIDLIST definition statement, an installation can specify the following information for VTAM:

- The identification character string for each BSC or TWX terminal on a switched line to be verified by VTAM.
- The symbolic name to be assigned to the terminal entering a verified identification character string. A unique name can be specified for each verifiable character string.

If VTAM is unable to verify a terminal identification, it disconnects the terminal or passes it to the application program designated to handle unidentified terminals. VTAM only disconnects the terminal if it cannot pass the terminal to an application program. An application program is designated to handle unidentified terminals by supplying special operands on a TERMINAL statement in the NCP definition deck. These special operands indicate:

- That the TERMINAL statement contains terminal descriptions to be applied to unidentified terminals. (This statement contains the CTERM=YES parameter and the UTERM specification.)

- That whenever such a terminal description is applied to a terminal, that terminal should be automatically logged onto a specified application program.

Any further verification of the terminal is then the responsibility of that application program. See “Network Control Program Requirements,” earlier in this chapter, for more information on the `TERMINAL` statement, including discussions on `CTERM=YES` and on `UTERM`.

To accomplish identification verification in a VTAM system under DOS/VS, the `IDLIST` definition statement should not be specified, but the `VIDLIST` definition statement should be coded. Thus, all identification character strings would be passed to VTAM for verification.

To accomplish identification verification in a VTAM system under OS/VS, three options are available:

- Use the method described for VTAM under DOS/VS; that is, perform all verification in the host CPU.
- Or specify `IDLIST` definition statements to describe all identifications to be verified by the NCP. Specify in an `IDLIST` definition statement that all unverified identifications are to be passed to VTAM. Code `VIDLIST` definition statements to handle the verification of all identifications not verified by the NCP.
- Or code only `IDLIST` definition statements, verifying all identifications in the communications controller and pass no unverified identifications to VTAM.

Verifying all identifications in the host CPU relieves the NCP of verification requirements and reduces space requirements of the NCP. On the other hand, it does increase VTAM requirements for virtual storage and for CPU time. This option is probably best suited for systems having only a small number of terminals to be verified and expecting only infrequent verification activity. It is also suited for systems whose NCP space and performance requirements are significantly more important than those of the CPU and VTAM.

Dividing the verification responsibility between the NCP and VTAM offers the greatest flexibility. The exact division can be weighed against space requirements, processing-time considerations, and verification usage.

Verifying identifications only in the NCP reduces VTAM and CPU requirements, and can free the access method of some processing requirements. This option might be selected for those systems expecting heavy identification-verification processing.

### ***Controlling Connections***

Before a terminal and an application program can communicate with each other in a VTAM system, they must be connected. VTAM provides the following facilities to assist an installation in controlling connections:

- An exit to allow an installation-coded routine to authorize connection dynamically requests during program execution.
- A mechanism to allow an installation to control which applications can acquire a terminal.
- The capability to tailor logon requests to installation requirements and specifications. This facility includes helping the installation to control which terminals and operators can log onto which application programs.

### **Authorization Exit-Routine**

VTAM allows an installation to code its own connection-authorization routine. The authorization exit-routine is executed each time a connection or a disconnection is

requested. The connection request can be a logon or an OPNDST macro instruction. The disconnection request takes the form of a CLSDST macro instruction.

Upon entry, this exit-routine is provided with the names of the nodes and the type of request. The exit-routine can then determine whether the request is valid.

Upon returning to VTAM, the authorization exit-routine indicates to VTAM whether the request should be permitted. If the request is valid, it is completed by VTAM; otherwise, it is rejected.

See “VTAM Definition,” in Chapter 3, for more information on this exit-routine.

## Acquiring and Passing Connections

An application program can request that a terminal be connected or queued for logon by using one of the following methods:

- OPNDST macro instruction with the ACQUIRE option.
- SIMLOGON macro instruction.
- CLSDST macro instruction with the PASS option.

In VTAM, control of these options have been given to the installation; that is, only application programs authorized by the installation during VTAM definition can issue these macro instructions. Application programs not authorized can only obtain connections by responding to logon requests with an OPNDST (with the ACCEPT option) macro instruction.

An installation specifies the authorization of an application program in the APPL definition statement. (See “VTAM Definition,” in Chapter 3, for details on the APPL statement.)

APPL authorization assists an installation in controlling which application programs can initiate connection requests. The following topic, “Controlling Logon Requests,” explains how an installation can control connection requests that originate outside an application program.

*Note:* These options can also be controlled through the authorization exit-routine; because they are all forms of connection, logon, or disconnection, these options cause the authorization exit-routine to be scheduled. See “Authorization Exit-Routine,” above.

## Controlling Logon Requests

Using VTAM definition procedures and the authorization exit, an installation can control which terminals can log onto which application programs. (Note: Since the remainder of the discussion on controlling logons treats the various types of VTAM logons, refer to “Characteristics of Logons,” in Chapter 3, for a description of logon types.)

To permit terminal-initiated logons in VTAM, an installation specifies the following during VTAM definition:

- For the terminal—In the GROUP, LINE, CLUSTER, VTERM, TERMINAL, or LU definition statement (for remotely attached terminals) and in the LOCAL definition statement (for locally attached terminals), specifies that a terminal is to be monitored by VTAM for logon requests and indicates the interpret table containing the logon description to be used when monitoring logon requests from that terminal.
- For the application program—In the APPL definition statement, specifies the name of the application program.

- For the logon message—Uses the INTAB, ENDINTAB, and LOGCHAR macro instructions to define interpret tables. Each interpret table contains descriptions of the valid logon messages that can be issued from each terminal and indicates which program is to receive the connection request for each message. (See “VTAM Definition,” earlier in this chapter, for details on these macro instructions.)

Using the logon information supplied by the installation via VTAM definition, VTAM’s logon facilities determine which logon requests should be routed to which application programs. In addition to controlling this determination, the user has two other opportunities to inspect and control the pending connection request. These opportunities are in the installation-coded authorization exit-routine and in the application program to which the logon request is directed.

The installation-coded authorization exit-routine is notified of all terminal-initiated logon requests. Upon entry, this exit-routine can make a further determination as to whether the connection is valid. If invalid, the connection request is rejected by VTAM; if valid, the request is passed to the application program. (See “VTAM Definition,” in Chapter 3, for a description of the authorization exit-routine.)

Finally, having passed VTAM’s logon facilities and the authorization exit-routine, the logon request must still be accepted by the application program before a connection is established. The application program can inspect the logon request and either accept the request or reject it. VTAM provides a number of tools for verifying logon requests in the application program. Using these tools effectively requires planning and preparation on the part of the installation at VTAM definition. It requires that specific procedures be established for the verification of logon requests in the application program, and it requires that application programs that can accept logon requests contain logon exit-routines adhering to these installation-established procedures. (Though an application program can accept a logon request without using a logon exit-routine, the program would have access to the logon message only through the exit-routine.)

For an application program to provide logon verification effectively, the installation should:

- Establish the format and content of permissible logon messages for each application program. Permissible messages might be required to contain terminal-user identifications and passwords.
- Use the INITAB, ENDINTAB, and LOGCHAR macro instructions to define these messages to VTAM. (The LOGCHAR macro instructions are also used to specify which application program is to receive notification for each logon request.)
- Use the VTERM, TERMINAL, LU, or LOCAL and the APPL definition statements to identify valid terminals and application programs to VTAM.
- Use the GROUP, LINE, CLUSTER, VTERM, TERMINAL, LU, or LOCAL definition statements also to specify which interpret table is to be used for each terminal.
- Modify, if necessary, and activate VTAM’s network solicitor (for start-stop and BSC terminals).

Once the installation has prepared the telecommunication system for terminal-initiated logons, the application program can provide a final authorization check. VTAM provides tools for use in the application program to assist in this final check. These tools include:

- A logon-exit in the application program—This exit is triggered whenever a logon request is to be passed to the application program.
- A means of obtaining the logon message—VTAM’s INQUIRE macro instruction can be used to obtain the logon message. If installation-defined procedures specify that the



message contain required information (such as a user identification and a password), this information can be inspected for accuracy and validated.

- A means of determining device type—VTAM's INQUIRE macro instruction can also be used to determine the type of device from which the message was received. If the application program is not equipped to handle such devices, the request can be rejected.
- A means of identifying the specific terminal—Input to the logon exit-routine includes the symbolic name of the terminal. The name is established by the installation at VTAM definition. Through installation-defined procedures, valid terminal names can be made known to each application program. Criteria for accepting or rejecting logon requests from specific terminals can then be established by the installation.

Thus, for a terminal to log onto an application program via VTAM's logon facilities, the following conditions must be met:

- The terminal, the logon message, and the application program must be defined by the installation at VTAM definition.
- An interpret table containing valid logon messages must be created for each terminal, unless the standard logon message is to be used.
- The installation-coded authorization exit-routine (if supplied) must authorize the logon request.
- The application program must accept the logon request.

VTAM also allows an installation to control other types of logons. As noted above, the automatic logon is specified at VTAM definition. Although this specification can be altered via a network-operator logon command, all connection requests can be monitored by the installation-coded authorization exit-routine. Also, the application program can accept or reject connection requests. Thus, even if an unauthorized connection is requested by the network operator, the request must still pass the authorization checking of the installation exit-routine and of the application program.

Application program logon requests are also controlled by the installation. (The application-program logon results from issuing a SIMLOGON or CLSDST, with the PASS option, macro instruction.) This logon request must be initiated by an application program, and it can only be initiated by authorized programs.

Authorization to issue the SIMLOGON or the CLSDST (with the PASS option) macro instructions is provided at VTAM definition via the application program's APPL definition statement. In addition to this authorization, connections resulting from application-program logon requests must pass the installation-coded authorization exit-routine check (if supplied) and must be accepted by the application program to be completed.

**Security Considerations for Standard Logon:** If a standard logon can be issued from a terminal, that terminal has access to any application program in the VTAM system. On the other hand, when an interpret table is specified for a terminal, logon requests from that terminal can only be accepted for application programs specified in the table.

If caution is not exercised when authorizing standard logons, the security of the telecommunication system may be jeopardized. As noted in "VTAM Definition," in Chapter 3, an installation can prescribe data to be added to the standard logon message. The application program can then inspect this data to decide whether to accept or reject the logon. In addition, a standard logon must also be validated by the installation-coded authorization exit-routine (if any). This routine could be designed to control security-

sensitive connections. Therefore, an installation, by requiring password data in the logon message and by coding an authorization exit-routine, can prohibit unauthorized use of the standard logon.

Caution should also be exercised when using the network-operator logon facility to log a terminal onto the network solicitor. If no interpret table is specified for such a terminal, a terminal operator might then have access to any application program in the VTAM system by using the standard logon.

### ***Symbolic Names***

As a further aid in controlling connections, VTAM nodes can only be addressed in the active system by symbolic names. The symbolic names are assigned at VTAM definition. Terminal names are provided via the `TERMINAL`, `VTERM`, `LU`, or `LOCAL` definition statements; NCP group names are provided via the `GROUP` definition statement; line names are provided via the `LINE` definition statement; cluster controller names are provided via the `CLUSTER` definition statement; component names are provided via the `COMP` definition statement; and application program names are provided via the `APPL` definition statement.

These names must be used by the network operator in addressing any nodes in a VTAM network. The application program name must be used by the application program (in the `OPEN` macro instruction, see "Controlling Access to VTAM," below) to access VTAM. The names assigned to terminals and components must be used also by the application program to initially access terminal nodes.

Thus, all nodes are symbolically named. Name assignments are controlled by the installation through VTAM definition facilities. These names must be used to address and access nodes within the VTAM network.

See "VTAM Node Structure," in Chapter 3, for more information on assigning symbolic names to nodes.

### ***Controlling Access to VTAM***

An application program must open a VTAM access method control block (ACB) before the program can use VTAM resources and facilities. To be opened, this control block must supply an application-program identification for verification by VTAM. The identification is the name of an `APPL` definition statement specified and filed in a member of the VTAM definition library during VTAM definition by the installation.

The installation may also specify in the `APPL` statement that any attempt to open an ACB using the name of the statement as an application-program identification must include a password. The password specified in the ACB must agree with that specified in the `APPL` definition statement if the ACB is to be permitted to open.

In addition to controlling the names of authorized application programs and the specification of passwords, an installation can control an application program's initial use of VTAM. A program's initial use of VTAM (that is, issuing an `OPEN` macro instruction for a VTAM ACB) can be restricted by controlling the activation of major nodes. Thus, even if the identification and the password match an `APPL` specification, the open request is denied if the major node containing the `APPL` specification has not been activated or if the specification is already in use by another opened ACB that used the same identification and password.

### ***Controlling the Use of VTAM Facilities***

Although an application program has been recognized by VTAM (a VTAM ACB has been successfully opened), it may still not have access to all of VTAM's facilities. Some of VTAM's facilities are considered sensitive resources. These facilities, if used without

discretion, could impact the integrity or security of the telecommunication system. They are therefore specifically identified, and their use is put under the control of the installation, via VTAM definition facilities.

These sensitive facilities include:

- The use of block processing instead of message or transmission processing (for terminals connected in basic-mode only).
- The passing of a terminal connection to another application program (CLSDST macro instruction with the PASS option).
- The acquiring of a terminal connection using the OPNDST macro instruction with the ACQUIRE option or the SIMLOGON macro instruction.

An application program can use block processing by specifying the BLOCK option of the PROC operand of the NIB macro instruction. See “Application Program Facilities,” in Chapter 5, for a description of block processing.

The other two facilities enable an application program to initiate a connection request for a terminal. VTAM attempts to establish and maintain the authorization of connections between terminals and application programs as the prerogative of the installation. To help ensure that only authorized connections are made, VTAM provides an installation control over application program-initiated connection requests.

An installation can control an application program’s use of sensitive facilities. This control is provided via the authorization specification in the APPL definition statement. Thus, in regard to block processing and application-program-initiated connection requests, an application program can use only those facilities specifically authorized in the APPL definition statement associated with its ACB.

### ***Protecting Sensitive Data***

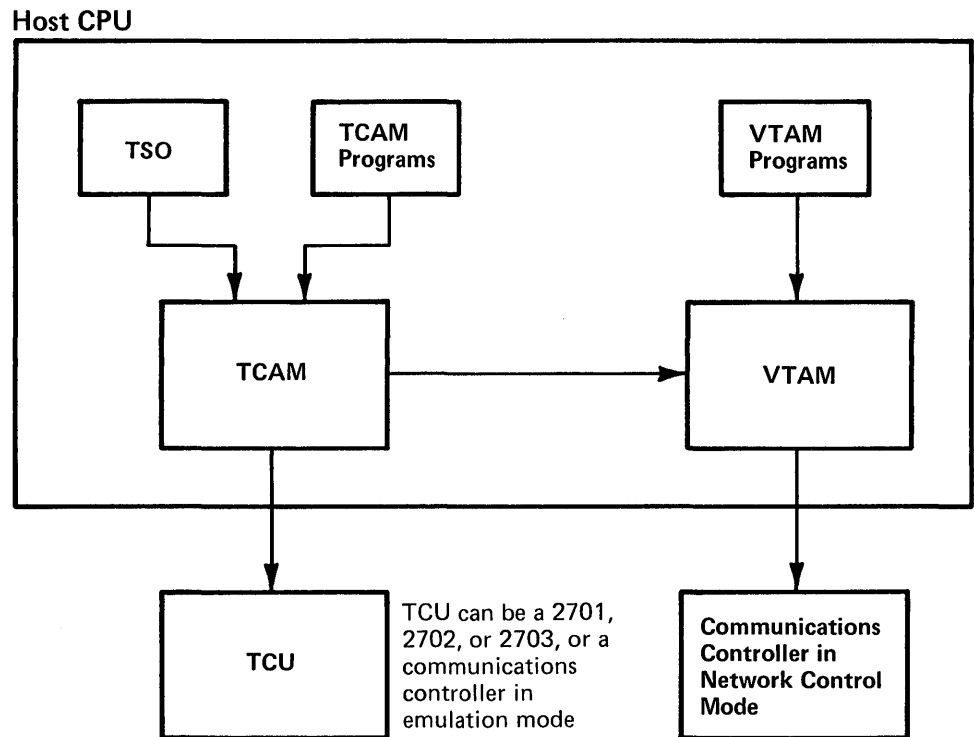
When data is transmitted between an application program and a terminal, it passes through VTAM and through NCP buffers. These buffers are obtained from and returned to common buffer pools for each transmission request. Buffers are cleared before they are allocated.

To protect sensitive data, the application program can request that VTAM and NCP buffers be cleared before being returned to the buffer pools. This request is made by specifying the PROC=CONFTEXT option of the node initialization block (NIB). (See “The VTAM Language,” in Chapter 5, for information on the NIB.)

Buffer traces of confidential data produces only the name of the application program, the name of the terminal, and the direction of the data flow. The actual data is not included in the trace records, as would be the case of a buffer trace of non-confidential data. The PROCT=CONFTEXT option is also used by VTAM’s trace facility to determine confidential data.

## **Other Telecommunication Access Methods**

VTAM can coexist with QTAM and BTAM under DOS/VS and with BTAM and TCAM under OS/VS. QTAM programs, BTAM programs, and TCAM programs that do not use the communications controller in network control mode can be executed concurrently as long as they have separate telecommunication networks. Additionally, when VTAM and TCAM are both in the operating system, TCAM programs that use terminals attached to a communications controller in network control mode are supported through VTAM. Figure 7-4 shows an OS/VS telecommunication system with TCAM and VTAM being executed concurrently.



VTAM and TCAM use VTAM to communicate with terminals in the VTAM network. TCAM programs, including TSO, can also communicate with terminals attached to a 2701, 2702, or 2703 transmission control unit, or through a communications controller in emulation mode.

Figure 7-4. Communications Controllers and Transmission Control Units in a Telecommunication Network

### *TCAM Programs Under VTAM*

TCAM application programs and the message control program (MCP) can share the resources of a VTAM telecommunication network with application programs written for VTAM. When sharing a network with TCAM, VTAM processes requests for all remote terminals attached to a communications controller in network control mode and, optionally, requests for all locally attached 3270s. TCAM supports terminals attached to other transmission control units, including those terminals attached to a communications controller in emulation mode (with or without PEP), and all locally attached devices other than the 3270s which are supported by VTAM. The TCAM user can choose between having individual local 3270 devices supported directly or through VTAM.

The principal advantages of this shared VTAM/TCAM capability are:

- The ability to share network resources between VTAM and TCAM programs.
- The ability to allow terminals to log onto TCAM.
- The availability of the queued-control capability of TCAM in a shared VTAM/TCAM system.

Existing TCAM application programs may not require changes, recompilation, or reassembly; their interface with the TCAM MCP remains the same. The TCAM MCP requires reassembly to gain access to the NCP under VTAM. However, application programs that use TCAM operator control commands should be reevaluated to ensure that they will operate as expected. When certain TCAM commands are issued from TCAM

application programs or authorized terminals and are directed to a terminal in the VTAM network (one attached to a communications controller in network control mode or a locally attached 3270), the command is not handled as in TCAM.

Most commands that control the communications controller and its network are provided by the network-operator facilities of VTAM. The TCAM commands that previously provided these functions are either rejected (for example, DUMP of the NCP) or provide a TCAM-only function without controlling the physical network (for example, SUSPXMITS stops TCAM transmission to a terminal but allows other VTAM application programs to send to it). As with previous versions of TCAM, the commands related to the 2701, 2702, and 2703 are rejected if directed to the NCP network.

The following TCAM macro instructions are altered slightly for VTAM operation; therefore, MCPs require reassembly: CODE, CUTOFF, DATETIME, INTRO, MSGFORM, MSGGEN, MSGLIMIT, SEQUENCE, STARTMH, and TERMINAL.

### *DOS/VS Coexistence*

In a DOS/VS system, QTAM, BTAM, and VTAM can operate concurrently. QTAM and BTAM programs do not interact with VTAM. QTAM programs use QTAM, and BTAM programs use BTAM to communicate with:

- Terminals attached to transmission control units.
- Terminals attached to communications controllers by line in emulation mode.
- Terminals attached locally (BTAM only).

VTAM application programs use VTAM to communicate with:

- Terminals attached to communications controllers by lines in network control mode.
- IBM 3270s attached locally.

Lines attached to communications controllers using the NCP with PEP can be used in either network control or emulation mode with an appropriate access method.

Figure 7-5 illustrates concurrent use of QTAM, BTAM, and VTAM in DOS/VS.

With concurrent execution of the access method, a single application program can use both BTAM and VTAM to communicate with separate networks, provided that all requirements of both access methods are met.

### *OS/VS Coexistence*

In an OS/VS system, BTAM, TCAM, and VTAM can operate concurrently.

BTAM programs use BTAM and TCAM programs use TCAM to communicate with:

- Terminals attached to transmission control units.
- Terminals attached to communications controllers by lines in emulation mode.
- Terminals locally attached.

TCAM programs use VTAM to communicate with:

- Terminals attached to communications controllers by line in network control mode.
- IBM 3270s locally attached.

*Note:* TCAM programs can communicate with locally attached 3270s either directly or through VTAM.

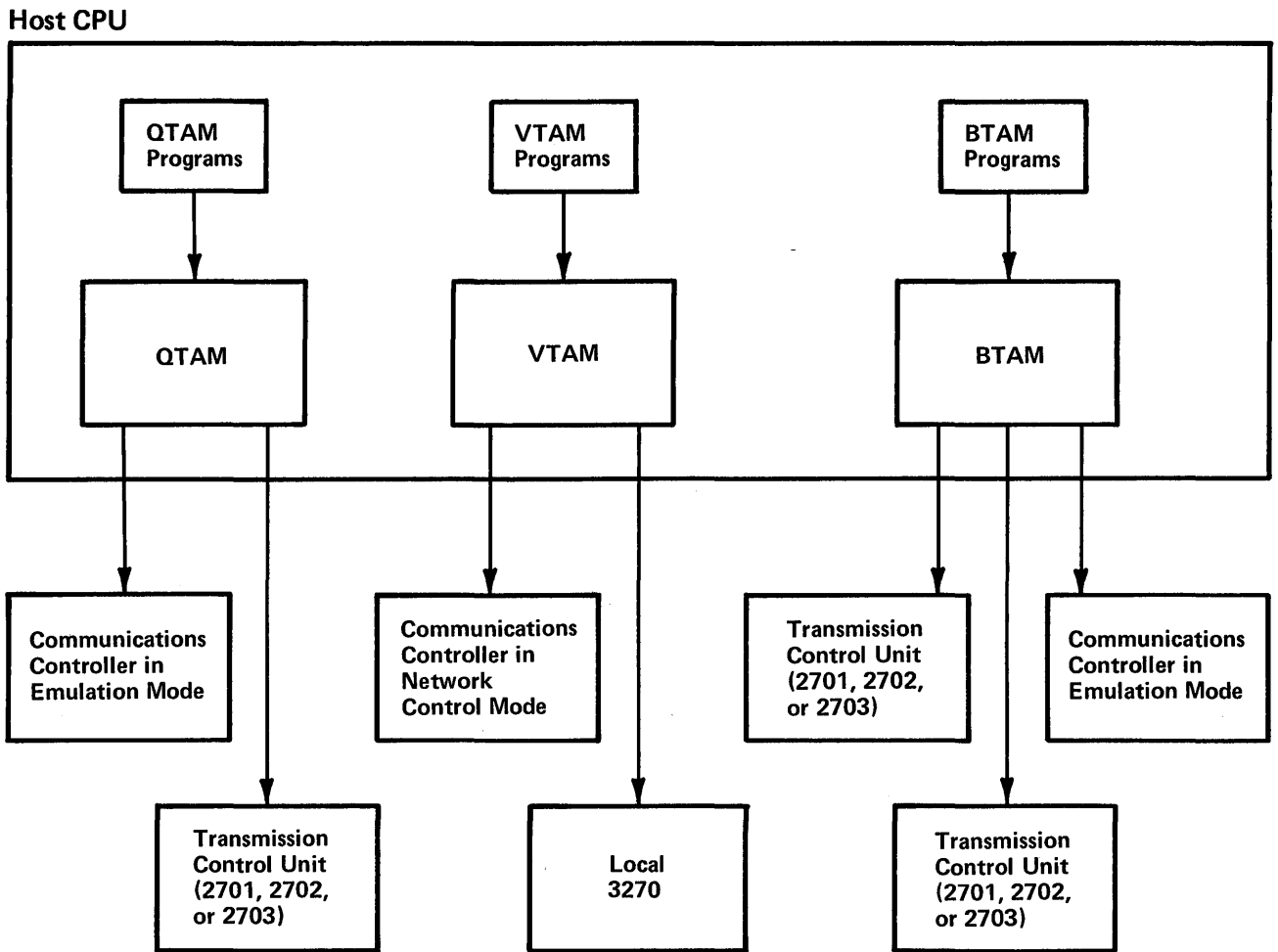


Figure 7-5. DOS/VS Coexistence

VTAM application programs use VTAM to communicate with:

- Terminals attached to communications controllers by line in network control mode.
- IBM 3270s locally attached.

Lines attached to communications controllers containing the NCP with PEP can be used in either network control or emulation mode with the appropriate access method.

Figure 7-6 illustrates the concurrent use of BTAM, TCAM, and VTAM in OS/VS.

With the concurrent execution of the access methods, a single application program can use both BTAM and VTAM to communicate with separate networks, provided that all of the requirements of both access methods are met. In addition, an application program can use both VTAM and TCAM; in this case, the networks can be separate or the same.

Host CPU

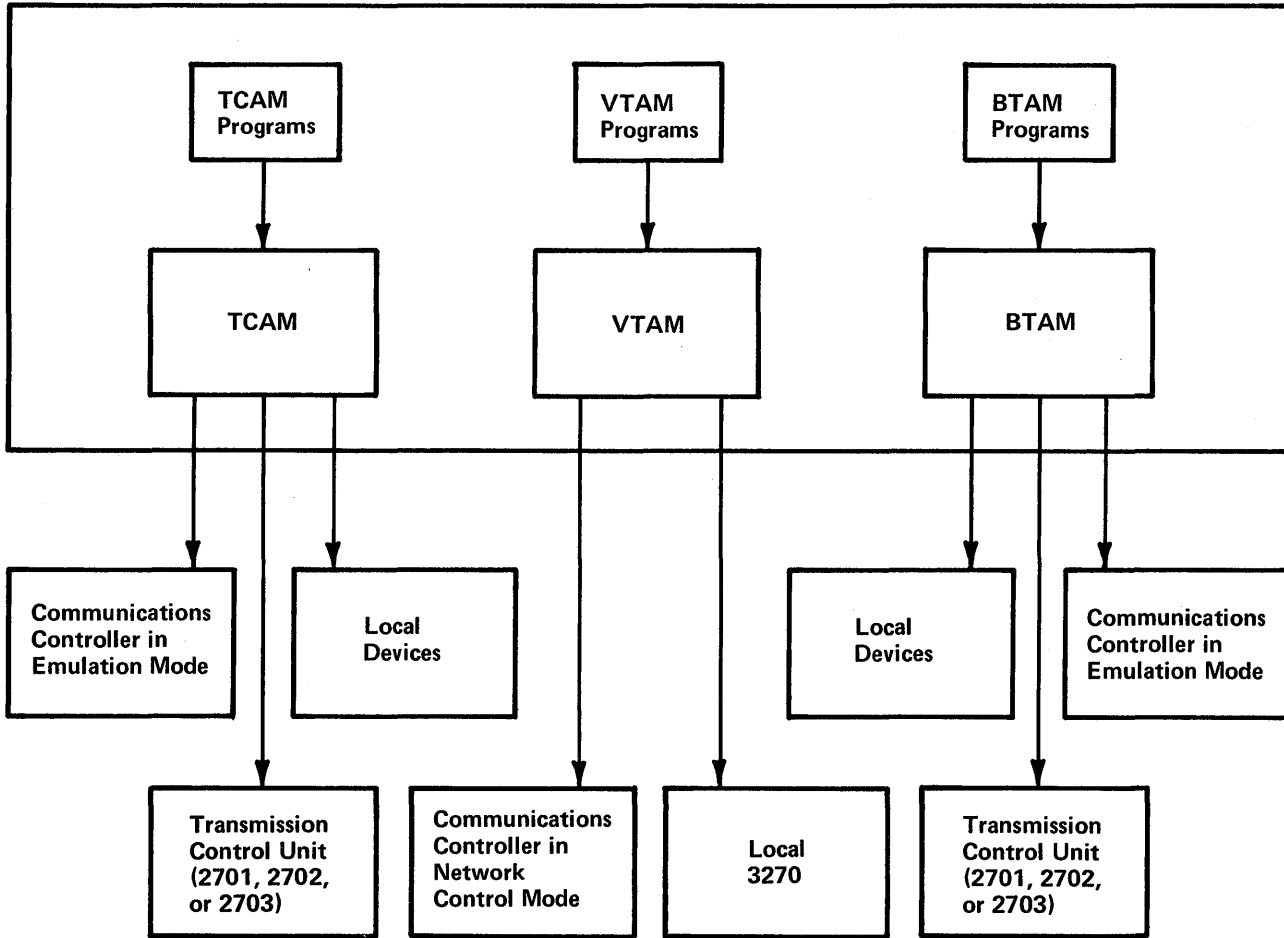


Figure 7-6. OS/VS Coexistence

## APPENDIX A. LOCAL 3270 SUPPORT LIST

This appendix details the devices and features supported by VTAM for locally attached 3270s.

Application programs can use either the basic-mode or the record-mode of VTAM to communicate with locally attached 3270s.

VTAM can be used with the following devices and features of the 3270 system:

3272 Control Unit (Models 1 and 2). Can include the attachment of 3277s, 3284s, and 3286s.

3277 Display Station (Models 1 and 2). Support is included for the following optional features:

- 6350—Selector Light Pen
- 9089—EBCDIC Character Set

3284 Printer (Models 1 and 2). Support is included for the following optional feature:

- 9089—EBCDIC Character Set

3286 Printer (Models 1 and 2). Support is included for the following optional feature:

- 9089—EBCDIC Character Set

Support is not included for the USASCII character set features for locally attached 3270s.





## APPENDIX B. REMOTELY ATTACHED TERMINALS

This appendix provides a detailed list of devices and their features supported as remote terminals by VTAM. Devices are listed by their line discipline: SDLC, start-stop, and BSC.

*Note:* VTAM receives and transmits data only in extended binary coded-decimal interchange code (EBCDIC). The NCP translates EBCDIC messages from VTAM to the appropriate transmission codes for remotely attached devices. The NCP also translates all messages in transmission codes, other than EBCDIC, to EBCDIC before sending them to VTAM.

### Synchronous Data Link Control (SDLC) Devices

SDLC devices can only be attached via a local communications controller in network control mode.

Each system described in this section is a teleprocessing subsystem for VTAM. A VTAM terminal for a teleprocessing subsystem is referred to as a logical unit. Application programs can communicate with logical units only through the record-mode of VTAM.

The devices listed below for each teleprocessing subsystem are treated by VTAM as SDLC cluster controllers.

*Note:* The SDLC cluster controllers cannot be operated in the same network containing start-stop or BSC terminals if that network is controlled by a 3704. Mixed operation of this type requires a 3705.

3600 Finance Communication System on nonswitched lines. The following devices in the 3600 system are supported in the VTAM network:

3601 Finance Communication Controller. Features of the 3601 and devices that attach to it are a function of the 3600 system, not of VTAM.

3614 Consumer Transaction Facility Models 1 and 2. Features of the 3614 are a function of the 3600 system, not of VTAM.

*Note:* The 3601 or 3614 cannot be operated with other devices in network control mode on a 3704. Mixed operation of this type requires a 3705.

3650 Retail Store System on nonswitched lines. The following device in the 3650 system is supported in the VTAM network:

3651 Store Controller (Model 50). Features of the 3651 (Model 50) and devices that attach to it are a function of the 3650 system, not of VTAM.

3660 Retail Store System on nonswitched lines. The following device in the 3660 system is supported in the VTAM network:

3651 Store Controller (Model 60). Features of 3651 (Model 60) and devices that attach to it are a function of the 3660 system, not of VTAM.

3790 Communication System on nonswitched lines. The following device in the 3790 system is supported in the VTAM network:

3791 Controller. Features of the 3791 and devices that attach to it are a function of the 3790 system, not of VTAM.

## Start-Stop Terminals

Start-stop terminals can be attached to either a local or a remote communications controller in network control mode.

Application programs can communicate with start-stop terminals only through the basic-mode of VTAM.

1050 Data Communication System on either switched or nonswitched lines. The following devices in the 1050 system are supported by VTAM:

1051 Control Unit (Model 1 or 2). The 1051 is required for attaching any of the devices listed below as part of the 1050 system. Support is included for any of the following 1051 optional features:

- 2953—Open Line Detection (World Trade only)
- 4795—Line Correction
- 4796—Line Correction Release
- 5465—Open Line Detection (U.S. only)

1052 Printer-Keyboard (Model 1 or 2). The 1052 requires a 1051. Support is included for the following 1052 optional features:

- 9567, 9597—PTTC/BCD Code
- 9571, 9591—PTTC/EBCD Code

1053 Printer (Model 1). The 1053 requires a 1051. Support is included for the following 1053 optional features:

- 9567, 9597—PTTC/BCD Code
- 9571, 9591—PTTC/EBCD Code

1054 Paper Tape Reader (Model 1). The 1054 requires a 1051.

1055 Paper Tape Punch (Model 1). The 1055 requires a 1051.

1056 Card Reader (Model 1 or 3). The 1056 requires a 1051.

1057 Card Punch (Model 1). The 1057 requires a 1051.

1058 Printing Card Punch (Model 1 or 2). The 1058 requires a 1051.

1092 Programmed Keyboard. The 1092 requires a 1051.

1093 Programmed Keyboard. The 1093 requires a 1051.

2740 Communication Terminal Model 1. Support is included for the following optional features for a 2740 on a switched line:

- 3255—Dial-up
- 6114—Record Checking
- 8028—Transmit Control

Support is included for the following optional features for a 2740 on a nonswitched line:

- 6114—Record Checking
- 7479—Station Control

Support is included for the following optional features for a 2740 on either a switched or a nonswitched line:

9567, 9597—PTTC/BCD Code  
9571, 9591—PTTC/EBCD Code  
Correspondence Code

*Note:* VTAM does not support the 2760 Attachment feature (8301).

System/7 Processor Station on switched or nonswitched lines. System/7 is supported as a 2740 Model 1 with Checking. VTAM requires:

1610—Asynchronous Communication Control

2740 Communication Terminal Model 2 on nonswitched lines. Support is included for the following 2740 optional features:

1495, 1496—Buffer Expansion  
1499—Buffer Receive  
6114—Record Checking  
9571, 9591—PTTC/EBCD Code

2741 Communication Terminal on either switched or nonswitched lines. Support is included for the following 2741 optional features:

3255—Dial-up (U.S. only)  
4708—Receive Interrupt  
7900—Transmit Interrupt  
9567, 9597—PTTC/BCD Code  
9571, 9591—PTTC/EBCD Code  
Correspondence Code

Communicating Magnetic Card SELECTRIC ® Typewriter on switched lines. This device is supported as a 2741 terminal on switched line with correspondence code. Thus, it is supported by VTAM as if it were a 2741 Communications Terminal equipped with a standard correspondence keyboard and element, the Dial-up feature, the Receive Interrupt feature, and the Transmit Interrupt feature.

World Trade Telegraph Terminals on nonswitched lines. Support is included for the following World Trade Telegraph optional features:

Paper tape reader  
World Trade TTY ZSC3 Figures Protected Code  
World Trade TTY ITA2 International Telegraph Alphabet 2

AT&T 83B3 Selective Calling Stations on nonswitched lines. Support is included for the following AT&T 83B3 optional feature:

Paper Tape Reader

WU Plan 115A Outstations on nonswitched lines. Support is included for the following WU 115A optional feature:

Paper Tape Reader

AT&T Teletypewriter TWX Terminal (Model 33 or 35) on switched lines. Support is included for the following AT&T TWX optional features:

- Data Interchange Code (8 level)
- Even Parity
- Forced Parity

## Binary Synchronous Communications (BSC) Terminals

BSC terminals can be attached to either a local or a remote communications controller in network control mode.

Application programs can communicate with BSC terminals, except for 3270s, only through the basic-mode of VTAM. Either the basic-mode or the record-mode of VTAM can be used to communicate with 3270s.

2770 Data Communication System on either switched or nonswitched lines. The following devices in 2770 system are supported by VTAM:

2772 Multipurpose Control Unit. The 2772 is required to support any of the other devices listed below as part of the 2770 system. Support is included for the following 2772 optional features:

- 1490—Buffer Expansion
- 1910—Conversational Mode
- 3250—Display Format Control
- 3650—EBCDIC Transparency
- 4610—Identification
- 5010—Multipoint Data Link Control
- 7950—Trans/Rec Monitor Print
- 9761—EBCDIC Code
- 9762—USASCII Code

50 Magnetic Data Inscrber. VTAM does not provide any editing of input from the 50. The 50 requires a 2772.

545 Output Punch (Model 3 or 4). The 545 requires a 2772.

1017 Paper Tape Reader (Model 1 or 2). The 1017 requires a 2772.

1018 Paper Tape Punch (Model 1). The 1018 requires a 2772.

1053 Printer (Model 1). The 1053 requires a 2772.

1255 Magnetic Character Reader. VTAM does not support 1255 Stacker Select. The 1255 requires a 2772.

2203 Printer (Model A1 or A2). The 2203 requires a 2772.

2213 Printer (Model 1 or 2). The 2213 requires a 2772.

2265 Display Station (Model 2). The 2265 requires a 2772.

2502 Card Reader (Model A1 or A2). The 2502 requires a 2772.

5496 Data Recorder. The 5496 requires a 2772.

2780 Data Transmission Terminal on either switched or nonswitched lines. Support is included for the following 2780 optional features:

- 5010—Multiple Record Transmission
- 5020—Multipoint Line Control
- 5820, 5821—Print Line
- 6400—Selective Character Set (USASCII)
- 7850—Terminal Identification
- 8030—EBCDIC Transparency
- 9761—USASCII Code
- 9762—EBCDIC Code

*Note:* VTAM does not support the 6-bit Transcode feature (9760).

2980 General Banking Terminal System on nonswitched lines. The 2980 system is supported by VTAM in the U.S. only. The following devices in the 2980 system are supported by VTAM:

2972 Station Control Unit (Model 8—RPQ 858160 and Model 11—RPQ 858231). Support is included for the following 2972 optional features:

- RPQ 835503—Buffer Expansion
- RPQ 858165, 858182—96-Character Buffer

2980 Teller Station (Model 1—RPQ 835504 and Model 4—RPQ 858147)

2980 Administrative Station (Model 2—RPQ 835505)

2971 Remote Control Unit (Model 3—RPQ 858144)

*Note:* VTAM does not support the Batched Message Input feature for 2980 stations.

3270 Information Display System on nonswitched lines. The following devices in the 3270 system are supported by VTAM:

3271 Control Unit (Model 1 or 2). A 3271 is required for attaching a 3277, 3284, or 3286. Support is included for the following 3271 optional features:

- 1550—Copy
- 9761—EBCDIC Code

3277 Display Station (Model 1 or 2). The 3277 requires a 3271. Support is included for the following optional features:

- 6350—Selector
- 9089—EBCDIC Character Set
- 9091, 9092—USASCII Character Set

3284 Printer (Model 1 or 2). The 3284 requires a 3271. Support is included for the following 3284 optional feature:

- 9089—EBCDIC Character Set

3286 Printer (Model 1 or 2). The 3286 requires a 3271. Support is included for the following 3286 optional feature:

- 9089—EBCDIC Character Set

3275 Display Station (Model 1 or 2). A 3275 is required for attaching a 3284. Support is included for the following 3275 optional features:

- 6350—Selector Light Pen
- 9089—EBCDIC Character Set
- 9761—EBCDIC Code

3284 Printer (Model 3). The 3284 requires a 3275. Support is included for the following 3284 optional feature:

- 9089—EBCDIC Character Set

System/3 Central Processing Unit on either switched or nonswitched lines. Support is included for the following System/3 optional features:

- 1315—Autocall (U.S. only)
- 2074—Binary Synchronous Communications Adapter
- 7477—Station Selection
- 7850—EBCDIC Transparency
- 9060—EBCDIC Code
- 9061—USASCII Code

System/7 Processor Station on either switched or nonswitched lines. System/7 is supported as a System/3 with the Binary Synchronous Communications Adapter (2074). IPL of System/7 or a nonswitched point-to-point line is not supported.

3740 Data Entry System on either switched or nonswitched lines. The 3740 is supported as a System/3. Thus, only 3740 functions equivalent to System/3 functions are supported by VTAM. The following devices in the 3740 system are supported by VTAM:

- 3741 Data Station (Model 2 or 4)

3747 Data Converter. Support is included for the following 3747 optional feature:

- 1660—Communications Adapter

3735 Programmable Buffered Terminal on either switched or nonswitched lines. Support is included for the following 3735 optional features:

- 5010—Multipoint Data Link Control
- 9761—EBCDIC Code
- 9762—USASCII Code

The following devices are also supported for the 3735:

- 5496 Data Recorder
- 3286 Printer (Model 3)

3780 Data Communications Terminal on either switched or nonswitched lines. If the 3780 does not include the card-punch component, the 3780 is supported as a 2770 without the Component Select feature. If the 3780 does include the card-punch component, the 3780 is supported as a 2770 with the Component Select feature. Thus only 3780 functions equivalent to 2770 functions are supported by VTAM. The following lists the supported optional features for the 3780:

- 3601—EBCDIC Transparency
- 5010—Multipoint Data Link Control

5701—Print Positions  
9761—EBCDIC Code  
9762—USASCII Code

**Note:** Space Compression/Expansion is not supported for the 3780.

System/370 as a remote station on switched or nonswitched lines. The System/370 CPU must be attached to a 2701, 2703, local communications controller in EP or network control mode, or an Integrated Communications Adaptor (ICA). Support is actually provided for the communications control unit as indicated in Figure C-2 in Appendix C. The following lists the supported optional features for each control unit:

2701 Data Adapter Unit

1302—Autocall (U.S. only)  
1303—Autocall (U.S. only)  
1314—Autocall (U.S. only)  
1355—Dual Code (U.S. only)  
3455—Dual Code (World Trade only)  
8029—Transparency  
9060—EBCDIC Code  
9061—USASCII Code

2703 Transmission Control Unit

1340—Autocall (U.S. only)  
1341—Autocall (U.S. only)  
7715—EBCDIC Code  
7716—USASCII Code

3704 Communications Controller—local

3705 Communications Controller—local

ICA for the System/370 Model 125

4640 Integrated Communications Adapter

ICA for the System/370 Model 135

4640—Integrated Communications Adapter  
9673—9680—Transparency  
9681—9688—USASCII Code

Figure B-1 is a chart that summarizes the support provided for each device in a VTAM network.



Device	TYPE OF NETWORK			Comments
	Point-to-Point Switched	Point-to-Point Nonswitched	Point-to-Point Nonswitched	
<i>SDLC</i> 3601 3614 3651 Models 50, 60 3791		X X X X	X X X X	SDLC terminals are logical units
<i>Local</i> 3270				Terminals are 3277, 3284, and 3286
<i>Start-Stop</i> 1050 2740 Model 1 2740 Model 2 2741 AT & T 33/35 TWX AT & T 83B3 & WU 115A Communicating Magnetic Card SELECTRIC® Typewriter System/7 World Trade Telegraph	X X X X X X X X	X X X X X X X X	X X X X X X X X	
<i>BSC</i> 2770 2780 2972 Models 8, 11 3270 (remote) 3735 3741 Models 2, 4; 3747 3780 System/3 System/7 System/370	(Note 1) X X X X X X X X X X X	X X X X X X X X X X X	(Note 2) X X X X X X X X X X X	Terminals are 2980 and 2971 Terminals are 3275, 3277, 3284, and 3286

**Notes:**

1. Except for the 3270 and the 2972, a BSC device can share a telephone number with any other BSC device in a point-to-point switched network.
2. Except for the System/7, a BSC device can share a multipoint nonswitched line with any other BSC device.

Figure B-1. Summary of Terminals That Can Operate in a VTAM Network

## APPENDIX C. REMOTE STATION VERSUS REMOTE CONTROLLER

Because VTAM supports the communications controller both remotely attached and as a remote station, a distinction should be made between the connections of locally attached and remotely attached communications controllers. In a VTAM network, there are two ways in which two communications controllers can be connected to each other. One way is to have a communications controller remotely attached as a satellite of a locally attached controller. The remote connection is depicted in Figure C-1. The other way is to connect two independently, locally attached communications controllers. The connection of locally attached communications controllers is depicted in Figure C-2.

In Figure C-1, both VTAM in the host CPU and the NCP in the locally attached communications controller control the remotely attached communications controller. They recognize and use the remote unit as a communications controller, and they communicate with it, directing its attached devices. All control for the remotely attached communications controller must emanate from the single host CPU and be passed through the locally attached communications controller.

Figure C-2 also represents a connection between two communications controllers, but in this case, neither controller is viewed as a remote communications controller by the other. Each controller with its attached host CPU (containing independently functioning VTAMs) is viewed by the other as a single terminal. In fact, there are two telecommunications networks almost totally independent of each other. In this type of connection, the two communications controllers treat each other as remote stations, not as a remote communications controller with the processing capability of an NCP.

In the network configuration depicted in Figure C-1, VTAM in CPU A can directly address any terminal attached to communications controller B. Figure C-2 depicts a network in which the VTAM in CPU A *cannot* directly address a terminal attached to communications controller B.

Tracing a message through the two systems will help to clarify the differences between the two types of attachments. Suppose an application program in CPU A is to send a message to terminal C:

*In the system depicted by Figure C-1:*

To send the message, the application program must be executing in the CPU with VTAM. The application program must request connection, using VTAM facilities, to terminal C. After the connection is complete, the application program must request VTAM to transmit the message to the terminal.

Upon receiving the request for data transmission, VTAM verifies that the terminal is in its network and transmits the message to the locally attached communications controller. This controller (A in the figure), in turn, determines that the request is for a terminal attached to the remote unit. Communications controller A then transmits the message to communications controller B. The remotely attached controller routes the message to the terminal.

Note that the application program need not be aware of how the terminal is attached. For example, if the terminal were a 3270, it could be attached to CPU A, to communications controller A, or to communications controller B. No matter what the attachment, the application program uses the same procedure to connect and communicate with the terminal.

*In the system depicted by Figure C-2:*

To send a message from an application program executing in the CPU in system A to a terminal (terminal C) in system B, the application program must be aware that terminal C is in another system. Also a companion application program must be executing in system B. The two application programs must be designed to work in coordination with each other.

Application program A first requests VTAM A for connection with the terminal that is system B. (Remember that each system is defined to the other as a single terminal.) The application program in CPU B also requests connections, but these requests are directed to VTAM B and are for terminal C and for the terminal that is system A.

To transmit the message, application program A requests VTAM A to transmit the message to the connected terminal (system B). VTAM A transmits the message to communications controller A which, in turn, writes the message to communications controller B. At this point, application program B must have issued a request to VTAM B for input from the terminal defined as system A. Upon receiving the message from controller A, communications controller B transmits it to VTAM B. VTAM B then sends the message to application program B. Using installation defined procedures, application program B determines that the message is destined for terminal C. So using VTAM B and communications controller B, application program B causes the message to be written to terminal C.

Thus when a communications controller is remotely attached (via a duplex line), its facilities and attached terminals are available directly to a single operating system. When two locally attached communications controllers are attached to each other, two operating systems are employed. Each operating system has direct access to the facilities and attached terminals of only its own communication controllers.

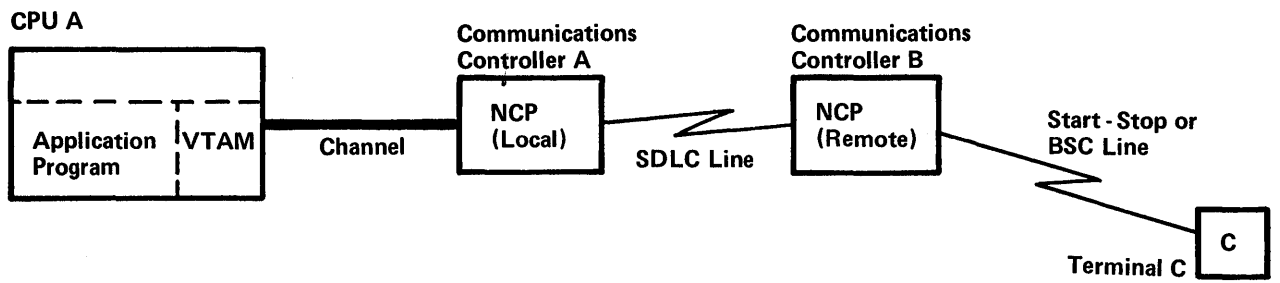
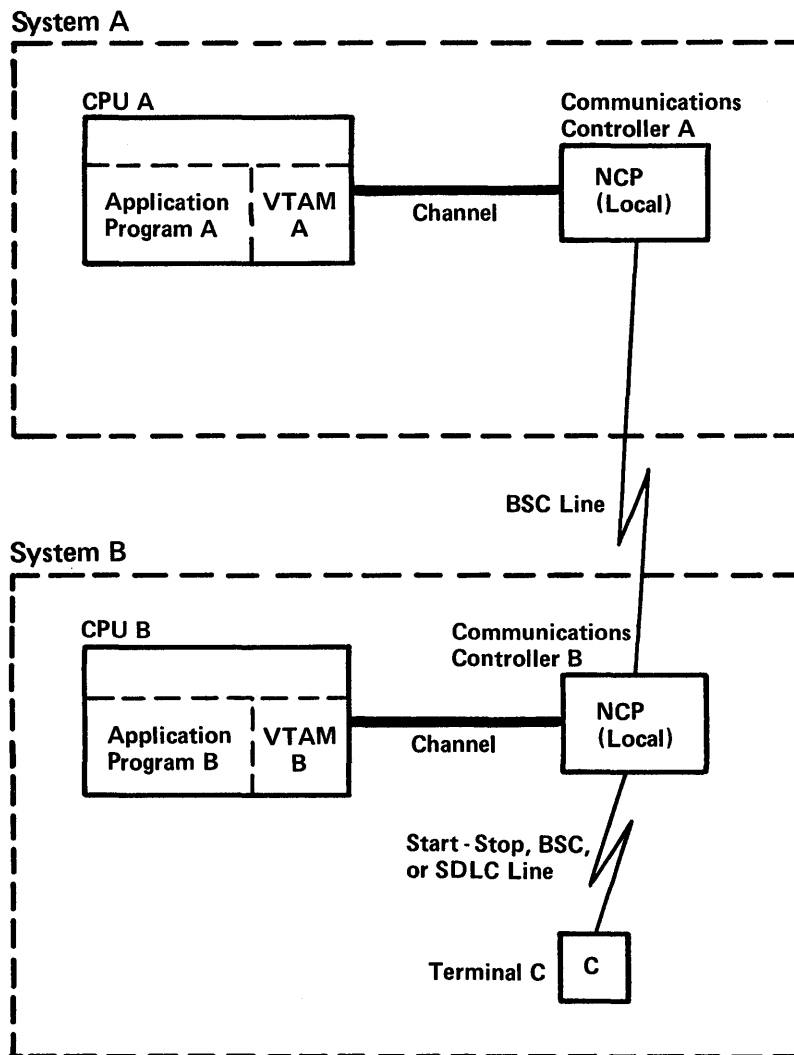


Figure C-1. A Remotely Attached Communications Controller



*Note:* VTAM does not have to be the access method in both CPUs to use a controller as a remote station. For example, if another telecommunication access method were in CPU B, System B would still remain a remote station for VTAM A.

Figure C-2. Communications Controllers Attached as Part of Remote Stations



## APPENDIX D. SUMMARY OF APPLICATION-PROGRAM INDICATORS

This appendix provides a summary of indicators that can be exchanged between an application program using VTAM and a logical unit. The indicators in this appendix are those that can be included as a message or part of a message.

A message can include data and/or indicators. A response can include:

- The type of response, that is FME or RRN and exception or normal.
- An explanation of the response if it is an exception.
- Change direction and bracket indicators. (These indicators can also be sent on messages and are described below.)

The remainder of this appendix is a table of the message indicators. The table is divided into columns; each column is defined as follows:

### Type of Indicator

specifies the name of the indicator. If the indicator has an abbreviation, the abbreviation is in parenthesis.

### Function

describes the use of the indicator.

### VTAM Application Program can SEND/RECEIVE

indicates whether the indicator can be transmitted or received by the application.

### Macro Used or RPL Field Set

indicates the VTAM macro instruction or RPL field used in specifying the indicator.

### Data-Flow Type

indicates whether the indicator is sent synchronously (DFSYN) or asynchronously (DFASY).

### Next Action Expected

indicates what action should occur following the exchange of indicator.

This table contains the following types of indicators:

- Synchronous-Flow.
- Asynchronous-Flow.
- Change-Direction.
- Bracket.
- SESSIONC.

Type of Indicator	Function	VTAM Application Can Send/Receive	Macro Used or RPL Field Set	Data-Flow Type	Next Action Expected
-------------------	----------	-----------------------------------	-----------------------------	----------------	----------------------

#### Synchronous-Flow Indicators

Bid	Bid to begin a bracket	Send only	SEND with CONTROL=BID	DFSYN	Receive response that says whether or not a bracket can be started
Cancel	Tell receiver to purge chain elements already received	Send	SEND with CONTROL=CANCEL	DFSYN	Response indicates cancellation
		Receive	CANCEL set in CONTROL field	DFSYN	Purge chain elements
Chase	Determine that the receiver has no more responses to send	Send	SEND with CONTROL=CHASE	DFSYN	When a response to the CHASE is received, all responses are accounted for
		Receive	CHASE set in CONTROL field	DFSYN	Send a positive or exception response
Logical Unit Status (LUS)	Inform receiver of an unexpected condition	Send	SEND with CONTROL=LUS	DFSYN	Varies with condition
		Receive	LUS set in CONTROL field	DFSYN	Varies with condition
Quiesce Complete (QC)	Tell the receiver that the sender is now quiesced and will not send until released	Send	SEND with CONTROL=QC	DFSYN	Await release-quiesce indicator
		Receive	QC set in CONTROL field	DFSYN	Send to quiesced receiver
Ready to Receive (RTR)	Tell the VTAM application program that a bracket has ended and a request to begin one can be sent	Receive only	RTR set in CONTROL field	DFSYN	Send a message that includes BRACKET=BB

#### Asynchronous-Flow Indicators

Quiesce-at-End-of-Chain (QEC)	Tell the receiver to quit sending now or, if chaining, at the end of this chain	Send	SEND with CONTROL=QEC	DFASY	Receiver should send a quiesce-complete indicator
		Receive	QEC set in the CONTROL field	DFASY	Finish sending, then send quiesce-complete
Release Quiesce (RELQ)	Permit the receiver to send now	Send	SEND with CONTROL=RELQ	DFASY	Expect message from receiver
		Receive	RELQ set in CONTROL field	DFASY	Send a message, if desired

Type of Indicator	Function	VTAM Application Can Send/Receive	Macro Used or RPL Field Set	Data-Flow Type	Next Action Expected
-------------------	----------	-----------------------------------	-----------------------------	----------------	----------------------

**Asynchronous-Flow Indicators (Cont.)**

Request Shutdown (RSHUTD)	Request the VTAM application program to send a SHUTD or to disconnect the logical unit	Receive only	RSHUTD set in CONTROL field	DFASY	Send a SHUTD or disconnect
Shutdown Complete (SHUTC)	Tel the receiver that preparation for shutdown is complete	Receive only	SHUTC set in CONTROL field	DFASY	Disconnect the logical unit
Shutdown (SHUTD)	Tell a logical unit to prepare to shut down	Send only	SEND with CONTROL=SHUTD	DFASY	The logical unit should send a SHUTC
Signal	Pass a four-byte message with an agreed-upon meaning	Receive only	SIGNAL set in in CONTROL field; value in SIGNAL field	DFASY	Installation-defined

**Change-direction Indicators**

The change-direction command indicator can be sent in a message that contains data, a synchronous-flow indicator, or a bracket indicator. It can also be sent in a response which, optionally, can include a bracket indicator.

The change-direction request indicator can be sent in a message that contains a QEC, RELQ, SIGNAL, or bracket indicator. It can also be sent in a response which, optionally, can include a bracket indicator.

Change-Direction Command (CMD)	Tell the receiver that it is now his turn to send	Send	SEND with CHNGDIR=CMD	DFSYN, DFASY, or RESP	Receive from the logical unit
		Receive	CMD in CHNGDIR field	DFSYN, DFASY, or RESP	Send to the logical unit
Change-Direction Request (REQ)	Request the receiver to send a change-direction command	Send	SEND with CHNGDIR=REQ	DFASY or RESP	Check incoming messages or responses for CMD in CHNGDIR field
		Receive	REQ in CHNGDIR field	DFASY or RESP	Send CMD when able



Type of Indicator	Function	VTAM Application Can Send/Receive	Macro Used or RPL Field Set	Data-Flow Type	Next Action Expected
-------------------	----------	-----------------------------------	-----------------------------	----------------	----------------------

### Bracket Indicators

The synchronous-flow indicators, bid and ready-to-receive, are used by the VTAM application program to determine whether it can send a begin-bracket indicator.

The bracket indicators can be sent in a message that contains data or a synchronous-flow indicator (except bid or ready-to-receive). A change-direction indicator can also be sent in the same message or response.

Begin Bracket (BB)	Begin a bracket	Send	SEND with BRACKET=BB	DFSYN	Receive response that indicates a bracket was begun
		Receive	BB set in BRACKET field	DFSYN	None
End Bracket (EB)	End a bracket	Send	SEND with BRACKET=EB	DFSYN	Receive response that indicates the bracket was ended
		Receive	EB set in BRACKET field	DFSYN	Can mean "end-of-transaction"

### SESSIONC Indicators

These indicators control and are sent separately from synchronous- and asynchronous-flow messages and their responses.

Bracket and change direction indicators cannot be sent with SESSIONC indicator messages or responses.

Clear	Stop all sending of synchronous- or asynchronous-flow messages and responses and clear pending ones from the network	Send only	SESSIONC with CONTROL=CLEAR	Not Applicable	Perform sequence number recovery
Request Recovery (RQR)	Tell the VTAM application program that a recovery procedure is required	Receive only	RQR set in CONTROL field in SCIP exit-routine	Not Applicable	Clear and then perform sequence number recovery
Start Data Traffic (SDT)	Allow the sending of synchronous- or asynchronous-flow messages and responses	Send only	SESSIONC with CONTROL=SDT	Not Applicable	Normal message flow can resume
Set-and-Test Sequence Number (STSN)	Exchange information with the logical unit so that sequence numbers can be determined and/or reset	Send only	SESSIONC with CONTROL=STSN and settings in IBSQAC and/or OBSQAC fields and IBSQVAL and/or OBSQVAL fields	Not Applicable	Receive response indicating logical unit reaction to STSN message

# GLOSSARY

This glossary defines all new VTAM terms and abbreviations used in this publication. It does not include all terms previously established for teleprocessing, DOS/VS, and OS/VS.

## A

**ACB:** Access method control block.

**accepting:** The process of connecting a node in response to a logon request from that node. A node is "accepted" with an OPNDST macro instruction having the ACCEPT option code set in its RPL.

**access method control block (ACB):** In VTAM, a control block that links an application program to VTAM.

**accounting exit-routine:** An installation-coded routine invoked by VTAM to collect statistics on the use of the telecommunication system.

**acquiring:** The processing of initiating and securing connection to another node. A node is "acquired" with an OPNDST macro instruction having the ACQUIRE option code set in its RPL.

**active:** Pertaining to a node that is connected to, or is available for connection to, another node. Contrast with *inactive*.

**any-mode:** (1) the form or READ or RECEIVE operation that obtains data from any single terminal, (2) the form of SOLICIT operation that solicits data from all eligible connected terminals, or (3) the form of OPNDST operation that establishes connection with any single eligible terminal from which a logon request has been received. The any-mode is established by specifying OPTCD=ANY for the RPL used by the READ, RECEIVE, SOLICIT, or OPNDST macro instruction.

**application program:** (1) To *VTAM*, the requests and control blocks that refer to a given ACB, or are pointed to by that ACB. (2) To the *programmer*, the program code that performs a given function. This code includes the code that processes the input and output data. It could include more than one ACB. (3) To the *operating system*, the program code that has been initiated as a job step or task.

For example: A given ACB may be identified to VTAM as PAYROLL1.

To VTAM, all requests and control blocks associated with the PAYROLL1 ACB constitute an application program. The programmer considers these requests and control blocks, and the program code needed to support these requests and to build and manipulate these control blocks, as the application program PAYROLL1. The programmer could use common supporting code for several ACBs—PAYROLL1, PAYROLL2, and PAYROLL3, for example. The programmer might think of these three as one application program or perhaps as three. VTAM would consider them as three separate application programs. If these three were combined into one job step, they would clearly be one application program to the operating system.

**application-program logon:** A logon request issued by an application program on behalf of a terminal. An application program must be authorized during VTAM definition to issue this type of logon.

**asynchronous flow messages:** Messages that are received ahead of any *synchronous flow messages* that might be queued for the application program or terminal.

For example: If an application program were to issue a RECEIVE macro instruction indicating that either synchronous or asynchronous flow messages could satisfy the macro instruction, VTAM would not satisfy the RECEIVE macro with synchronous flow messages until it had determined that no asynchronous flow messages were available.

**asynchronous request:** A request that causes control to be returned to the application program *before* the requested operation is completed. When the operation is completed, VTAM either invokes the RPL exit-routine, or posts an ECB. A request is made asynchronous by setting the ASY option code in its RPL. Contrast with *synchronous request*.

**authorization exit-routine:** An installation-coded routine that is invoked by VTAM for each connection and disconnection request to determine whether the request should be processed.

**automatic logon:** A logon request to a specified application program, generated by VTAM (rather than by the terminal itself) when the terminal becomes available for connection and the application program has opened its ACB and issued SETLOGON (OPTCD=START). Automatic logon is specified by the installation during VTAM definition.

## B

**basic-mode:** A set of facilities (including the macro instructions needed to use them) that enable the application program to communicate with BSC and start-stop terminals, including the locally attached 3270 Information Display System. READ, WRITE, SOLICIT, RESET, DO, and LDO macro instructions are basic-mode macro instructions.

**bid indicator:** An indicator used to determine if a new bracket can be started. The node receiving the bid indicator sends a normal response if a new bracket can be started or sends an exception response if a new bracket cannot be started. A bid indicator is sent when a SEND macro instruction is issued with CONTROL=BID set in its RPL.

**block:** In VTAM, the smallest unit of data that may be transmitted between an application program and a terminal connected in basic-mode. The maximum size of a block is determined by the characteristics of the device that is sending or receiving the data. For start-stop devices, a block is a unit of data beginning with an EOA or EOB character, and ending with an EOT or EOB character; for BSC devices, a block is a unit of data between an STX or SOH character and an ETB or ETX character. Contrast with *message* and *transmission*.

**bracket:** An exchange of data between an application program and a logical unit which accomplishes some task.

**bracket communication:** A method of communication in which a node does not begin a new bracket until the current bracket has been completed.

**BSC:** Binary synchronous communications.

## C

**CA mode:** See *continue-any mode*.

**cancel indicator:** An indicator that signifies to its receiver that the current chain being received should be discarded. A cancel indicator is sent when a SEND macro instruction is issued with CONTROL=CANCEL set in its RPL.

**change-direction communication:** A method of communication in which the transmitting node ceases transmitting on its own initiative, signals this fact to the other node, and prepares to receive.

**change-direction-request indicator:** An indicator sent by one node to another requesting that a *change-direction-command indicator* be returned.

**change-direction-command indicator:** An indicator sent by one node to another indicating that the sending node has finished transmitting and is prepared to receive.

**chase indicator:** An indicator that when returned to its originator, signifies all responses have transmitted. A chase indicator is sent by issuing a SEND macro instruction with CONTROL=CHASE set in its RPL.

**CID:** *communications identifier.*

**clear indicator:** A SESSIONC indicator sent by one node to another that prevents the exchange of messages and responses.

**closedown:** The process of deactivating VTAM and the telecommunication network. See also *quick closedown* and *orderly closedown*.

**cluster control unit:** A device that can control the input/output operations of more than one device. A remote cluster control unit can be attached to a *host CPU* only via a *communications controller*. A cluster control unit may be controlled by a program stored and executed in the unit; for example, the IBM 3601 Finance Communications Controller. Or it may be controlled entirely by hardware; for example, the IBM 2972 Station Control Unit. See also *communications controller* and *SDLC cluster controller*.

**communication control unit:** A general term for a communication device that controls the transmission of data over lines in a telecommunication network. Communication control units include transmission control units and communications controllers.

**communication line:** Any physical link, such as a wire or a telephone circuit, that connects one or more remote terminals to a communication control unit, or connects one communication control unit with another.

**communications controller:** A type of communication control unit whose operations are controlled by a program stored and executed in the unit. Examples are the IBM 3704 and 3705 Communications Controllers.

**communications identifier:** A VTAM equivalent for a terminal's symbolic name. The installation assigns a symbolic name to each terminal (or dial-up line) in its network configuration. When the application program requests connection to the terminal—by placing the terminal's symbolic name into a NIB and issuing an OPNDST macro instruction—VTAM converts this eight-byte symbolic name into a four-byte CID (communications ID). The CID is placed in the NIB and in the RPL used in the connection request. The application program must use this CID for all subsequent communication requests for the terminal.

**configuration restart:** In VTAM, the facility for reloading an NCP automatically following a failure in the communications controller containing that NCP. Configuration restart includes the ability to restore the network to its status just prior to the failure.

**connection:** In VTAM, in response to a request from an application program, the linking of VTAM control blocks in such a way that the program can communicate with a particular terminal. The connection process includes establishing and

preparing the network path between the program and the terminal. Contrast with *queued for logon*.

**continue-any (CA) mode:** A state into which a terminal is placed that allows its input to satisfy an input request issued in the any-mode. While this state exists, input from the terminal can also satisfy input requests issued in the specific-mode. (Contrast with *continue-specific* mode, where input from the terminal can satisfy *only* input requests issued in the specific-mode.) Continue-any mode is established by specifying OPTCD=CA for the RPL used by an OPNDST or any I/O macro instruction.

**continue-specific (CS) mode:** A state into which a terminal is placed that allows its input to satisfy only input requests issued in the specific mode. Continue-specific mode is established by specifying OPTCD=CS for the RPL used by an OPNDST or any I/O macro instruction.

**conversational write operation:** A composite operation wherein data is first sent to a terminal, and data is then read from that terminal. It is implemented with a WRITE macro instruction having the CONV option code set in its RPL.

**CS mode:** See *continue-specific mode*.

## D

**data transfer:** In telecommunications, the sending of data from one node to another.

**data transmission:** Same as *data transfer*.

**definition statement:** The means of describing an element of the telecommunication system to VTAM.

**device-dependent:** A characteristic of VTAM such that the application program is responsible for controlling the terminal to which it is connected. The application program is not responsible for controlling the use of the line by which the terminal is attached.

**direct-control:** A characteristic of VTAM such that an application program and a terminal must both be active and connected to each other before communication is possible.

**disconnection:** In VTAM, the disassociation of VTAM control blocks in such a way as to end communication between the program and a connected terminal. The disconnection process includes suspending the use of the network path between the program and the terminal.

## E

**emulation mode:** The functions of an *NCP* that enable it to emulate a transmission control unit. Contrast with *network control mode*.

**error lock:** A condition established by the communications controller wherein communications with the terminal is suspended.

**exception message:** A message that represents another message that never arrived, or for which a transmission error occurred. *Exception* messages are not sent by VTAM application programs or terminals; *messages* are sent which either arrive as normal messages or are replaced with exception messages. Upon receiving an exception message, the application program or terminal usually returns an *exception response*.

**exception response:** A response sent by a terminal or application program indicating that a particular message did not arrive normally.

**exit list:** In VTAM, a control block that contains the names of routines that receive control when specified events occur during VTAM execution. For example, programs named in the exit list handle such conditions as logon processing or I/O errors. Abbreviated *EXLST*.

**exit list routine:** A routine whose address has been placed in an exit list (EXLST) control block. The addresses are placed there with the EXLST macro instruction, and the routines are named according to their corresponding operand: hence SYNAD exit list routine, LERAD exit list routine, DFASY exit list routine, and so forth. All exit list routines are coded by the application programmer. Contrast with *RPL exit-routine*.

**EXLST:** *Exit list*

## F

**FME response:** A response that indicates whether its associated message was or was not successfully forwarded to its final destination (such as the display screen of an output device).

## H

**host CPU:** The central processor for a VTAM telecommunication system. VTAM resides in the host CPU. Devices attached by channels to the host CPU are said to be local devices. Remote devices must be attached to the host CPU via a local communications controller.

## I

**inactive:** Pertaining to a node that is neither connected to nor available for connection to another node. Contrast with *active*.

**interpret table:** In VTAM, an installation-defined correlation list that translates an argument into a string of eight characters. Interpret tables can be used to translate a logon message into the name of an application program for which the logon request is intended.

## L

**leading graphics:** From one to seven graphic characters that may accompany an acknowledgment sent to or from a BSC terminal in response to the receipt of a block of data.

**local:** Pertaining to the attachment of devices directly by channels to a host CPU. Contrast with *remote*.

**logical connection terminal:** For an NCP in network control mode, a description of a terminal (provided by a TERMINAL statement) to be used for a dial-in terminal that cannot be identified.

**logical unit:** The combination of programming and hardware of a *teleprocessing subsystem* that comprises a *terminal* for VTAM.

**logoff:** A request from a terminal to be disconnected from an application program.

**logon:** In VTAM, a request by or on behalf of a terminal to be connected to an application program.

**logon-interpret routine:** In VTAM, an installation-coded exit-routine that assists in the translation of an argument for an interpret table.

**logon message:** In VTAM, the data that can accompany a logon request received by the application program to which the request is directed.

**logon request:** See *logon*.

## M

**major node:** A set of one or more minor nodes represented by a single symbolic name. A major node can be a set of local terminals, a set of application programs, or a network control program.

**message:** (1) For logical units, the unit of information (data or indicators) that is sent by an application program or logical unit on its own initiative. Contrast with *responses*, which are sent only in reply to messages that have been received. (2) For BSC devices, the data unit from the beginning of a *transmission* to the first ETX character, or between two ETX characters. For start/stop devices, "message" and "transmission" have the same meaning.

**minor node:** A node within a *major node* that can be addressed separately from the major node. Examples of minor nodes are a terminal, a terminal component, an application program, or a line.

## N

**NCP:** See *network control program*.

**NCP generation:** See *network control program generation*.

**network control mode:** The functions of an *NCP* that enable it to direct a communications controller to perform telecommunication activities such as polling, device addressing, dialing, and answering. Contrast with *emulation mode*.

**network control program:** A program, transmitted to and stored in a communications controller, that controls the operation of the communications controller. Abbreviated *NCP*.

**network control program generation:** The process, performed in a central processing unit, of assembling and link-editing a macro instruction program to produce a network control program.

**network definition:** The process of defining, to VTAM, the identities and characteristics of each node in the telecommunication system and the arrangement of the nodes in that system.

**network operator:** The person responsible for controlling the operation of the telecommunication network.

**network-operator command:** A command used by the network operator to monitor or control the telecommunication network.

**network operator console:** A system console from which a network operator controls a telecommunication system.

**network-operator logon:** A logon request issued in behalf of a terminal from the network-operator console by using a network-operator command.

**NIB:** *Node initialization block*.

**NIB list:** In VTAM, a series of contiguous NIBs (node initialization blocks).

**node:** A point in a telecommunication system defined to VTAM by a symbolic name. See also *major node* and *minor node*.

**node initialization block:** A control block, associated with a particular node that contains information used by the application program to identify a node and indicate how communication requests directed at the node are to be implemented. Abbreviated *NIB*.

**node name:** In VTAM, the symbolic name assigned to a node during network definition.

## O

**orderly closedown:** The orderly deactivation of VTAM and the telecommunication network. An orderly closedown does not take effect until all application programs have disconnected from VTAM. Until then, all data transfer operations continue. Contrast with *quick closedown*.

## P

**partitioned emulation programming extension:** A function of the NCP that enables a communications controller to operate some communication lines in network control mode while simultaneously operating other in emulation mode. Abbreviated PEP.

**path:** The intervening nodes connecting a VTAM terminal and an application program in the host CPU.

**PEP:** *partitioned emulation programming extension*.

## Q

**QC indicator:** See *quiesce-completed indicator*.

**QEC indicator:** See *quiesce-at-end-of-chain indicator*.

**queued connection request:** A connection request that is directed at a terminal that is currently connected to another application program and for which the issuer has specified that the request is not to be completed until the terminal becomes available.

**queued for logon:** In VTAM, the state of a terminal that has logged on to an application program but has not yet been accepted for connection by that application program. Contrast with *connection*.

**queued logon request:** A logon request that has been directed at an application program but not yet accepted by that application program.

**quick closedown:** In VTAM, a closedown in which current data-transfer operations are completed, while pending data-transfer requests and new connection and data-transfer requests are canceled. Contrast with *orderly closedown*.

**quiesce-at-end-of-chain indicator:** An indicator sent by one node to another indicating that the other node should stop transmitting synchronous-flow messages after it has sent the last record of the chain being transmitted. When the other node returns a QC indicator, it cannot again transmit synchronous-flow messages until a RELQ indicator is received from the first node. A QEC indicator is sent when a SEND macro instruction is issued with CONTROL=QEC set in its RPL.

**quiesce-completed (QC) indicator:** An indicator sent by a node indicating that it will not transmit synchronous-flow messages again until it receives a RELQ indicator from the other node. A QC indicator is sent when a SEND macro instruction is issued with CONTROL=QC set in its RPL.

**quiesce communication:** A method of communicating in one direction at a time. Using this method, either node can assume the exclusive right to send synchronous-flow messages by getting the other node to agree not to send such messages. When the quiescing node wants to receive, it can release the other node from its quiesced state, allowing that node to send.

**quiescing:** In a VTAM application program, a way for one node to stop another node from sending synchronous-flow messages. Quiescing requires the sending of a quiesce-at-end-of-chain indicator and can include the receiving of a quiesce-completed indicator. Sending by the quiesced node can be restarted by the

quiescing node sending a release-indicator. Among other reasons, quiescing can be used to: (1) ensure a communication pattern in which only one node can send at a time, (2) stop the continuous sending of data by one node because a buffer in the other node is about to overflow, or (3) occasionally interrupt continuous sending of data by one node so that output can be sent by the other node.

## R

**record:** The unit of data transmission for record-mode. A record represents whatever amount of data the transmitting node chooses to send.

**record-mode:** A set of facilities (and the macro instructions needed to use them) that enable the application program to communicate with logical units or with the locally- or remotely-attached 3270 Information Display System. SEND, RECEIVE, and RESETSR are record-mode macro instructions.

**release-quiesce (RELQ) indicator:** An indicator sent by a node indicating that the other node can begin transmitting synchronous-flow messages. A RELQ indicator is sent when a SEND macro instruction is issued with CONTROL=RELQ set in its RPL.

**RELQ indicator:** See *release-quiesce indicator*.

**remote:** Pertaining to the attachment of devices to a central computer through a communication control unit. Contrast with *local*.

**remote station:** In VTAM, a central processing unit with its attached devices that is part of the telecommunication network, attached to a communications controller via a start-stop or BSC line, and is treated as a terminal by VTAM.

**request parameter list:** A control block that contains the parameters necessary for processing a request for connection, communication, or a request for an operation related to connection or communication. Abbreviated RPL.

**responded output:** A type of output request that is completed when the logical unit receives the message and returns a response (if one is called for) for it. Responded output occurs if POST=RESP is specified for the RPL used by a SEND macro instruction.

**response:** The unit of information that is sent by an application program or terminal in reply to a message that has been received.

**RPL:** *request parameter list*.

**RPL exit-routine:** A routine indicated by an RPL, that is to be scheduled by VTAM when the request for which the RPL is being used has been processed.

**RRN response:** A response that indicates that the node sending the response has accepted recovery responsibility for the associated message.

## S

**scheduled output:** A type of output request that is completed (as far as the application program is concerned) when its output data area is free. Contrast with *responded output*. Scheduled output occurs if POST=SCHED is specified for the RPL used by a SEND macro instruction.

**SDLC:** Synchronous data link control.

**SDLC cluster controller:** A cluster control unit for a tele-processing subsystem.

**SDT indicator:** See *Start-data-traffic indicator*.

**sequence number:** 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, another for messages sent from the logical unit to the application program) increases by one for each successive message transmitted. The value increases by one throughout the life of the connection unless reset by the application program with an STSN signal.

**SESSIONC indicators:** Indicators that can be sent from one node to another without using SEND or RECEIVE macro instructions. SDT, clear, and STSN are SESSIONC indicators. All SESSIONC indicators are sent with a SESSIONC macro instruction.

**set-and-test-sequence-number (STSN) indicators:** A set of SESSIONC indicators sent by one node to another to establish the proper sequence number.

**shared:** Pertaining to the availability of a resource to more than one user at the same time. In VTAM, communication controllers and communication lines can be shared concurrently by several application programs to communicate with different nodes. In VTAM, terminals are shared consecutively; only one application program can be connected to a terminal at any one time.

**specific-mode:** (1) The form of READ, RECEIVE, or SOLICIT operation that obtains data from one specific terminal, or (2) the form of OPNDST operation that establishes connection with one specified terminal if (or when) a logon request is received from that terminal. Specific mode is established by specifying OPTCD=SPEC for the RPL used by the READ, RECEIVE, SOLICIT, or OPNDST macro instruction.

**Start-data-traffic (SDT) indicator:** SESSIONC indicator sent by one node to another that enables data flow between them.

**STSN indicators:** See *set-and-test-sequence-number indicators*.

**synchronous-flow message:** Messages that can satisfy a RECEIVE macro instruction only if no *asynchronous-flow messages* are available to satisfy the macro instruction. (The RECEIVE macro instruction in this definition is one which can be satisfied by either type of message.)

**synchronous request:** A request that causes control to be returned to the application program only *after* the requested operation has been completed. A request is made synchronous by setting the SYN option code in its RPL. Contrast with *asynchronous request*.

## T

**TCU:** *transmission control unit*.

**telecommunication network:** In a telecommunication system, the combination of all terminals and other telecommunication devices and the lines that connect them.

**telecommunication system:** In a teleprocessing system, those devices and functions concerned with the transmission of data between the central processing system and the remotely located users. In VTAM, the telecommunication system includes the *host CPU*, *application programs* using VTAM, VTAM, the *telecommunication network*, and the channels that link the host CPU and the network.

**teleprocessing subsystem:** In VTAM, a secondary or subordinate network (and set of programs) that is part of a larger

teleprocessing system; for example, the combination consisting of an *SDLC cluster controller*, its stored program, and its attached input/output devices.

Examples of teleprocessing subsystems are the IBM 3600 Finance Communication System and the IBM 3650 Retail Store System.

**teleprocessing system:** The devices and functions of a data processing system that enable users at remote locations to access the data processing capabilities of a centrally located computer. A teleprocessing system has two major functions: the transmission of data between the central computer and the remote locations (performed by the telecommunication system) and the actual processing of the data in the central computer.

**terminal:** A node in a telecommunication network at which data can enter or leave the network. A terminal can be an input/output device, a terminal control unit to which one or more input/output devices (terminal components) are attached, a *logical unit*, or a *remote station*.

**terminal component:** A separately-addressable part of a terminal that performs an input or output function.

**terminal-initiated logon:** A logon request that originates from the terminal.

**transmission:** In telecommunications, a logical group of one or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. Contrast with *block* and *message*.

**transmission control unit:** A type of communication control unit whose operations are controlled solely by programmed instructions from the system to which the unit is attached; no program is stored or executed in the unit. Contrast with *communications controller*.

**transparent text mode:** A mode of binary synchronous transmission in which only line control characters preceded by DLE are acted upon as line control characters. All other bit patterns that happen to be line control characters are transmitted as data.

## V

**Virtual Telecommunications Access Method:** A set of IBM programs that control communication between terminals and application programs running under DOS/VS, OS/VS1, and OS/VS2.

**VTAM:** *Virtual Telecommunications Access Method*.

**VTAM definition:** The process of (1) including VTAM in the operating system generation (SYSGEN), (2) defining the teleprocessing network to VTAM and generating communication controllers' network control programs, and (3) modifying IBM-defined VTAM characteristics to suit the needs of the installation. VTAM definition is implemented by the installation with definition statements and operator commands.

**VTAM definition library:** The DOS/VS files or OS/VS data sets that contain the VTAM definition statements filed during VTAM definition. These statements describe the telecommunication system to VTAM and can be used to tailor VTAM and the system to suit the needs of the installation.

**VTAM load module library:** The library containing the VTAM load modules in OS/VS or VTAM phase in DOS/VS.



# INDEX

- abnormal termination 147
- ACB control block
  - description 111
  - relationship to executable macros and other control blocks 113
- accepting connection
  - general concept 82
  - to a specific logical unit 83
  - to any logical unit 83
  - to logical units 82
  - with LOGON exit-routine 82
- accounting-exit routine
  - coding and installing 40
  - considerations 41
  - input 41
- acquiring and passing connections and telecommunication security 172
- acquiring connection
  - general concept 83
  - types of acquisition 83
    - CONALL option 83
    - CONANY option 83
- acquiring logical units that are connected to another program 83
- activate-physical signal 127
- activating (see also "activating and deactivating")
  - major nodes, as defined options 30
  - minor nodes 44
  - nonexistent terminals 61
  - remote NCPs 44
- activating a node 55,30
- activating and deactivating
  - a remotely attached communications controller 58
  - an NCP 58
  - local 3270s 57
  - nodes 55
  - PEP 59,162
  - teleprocessing subsystems 61
- active
  - application program 61
  - terminal 61
- address of local 3270, specified on a LOCAL statement 27
- addressability 126
- AID (attention identifier) 136
- allocating
  - communications controllers 18
  - data sets 18
  - terminals 18
- allocation 5
- alternatives in writing a VTAM application program 140
- any-mode 101
  - compared with BTAM READ Initial 97
  - compared with specific mode 101
- APPL statement
  - contents 29
  - filing 29
  - name
    - example in sample program 18
    - in a VTAM application program 114
- application program
  - active 61
  - as an ACB 9
  - authorization and its effect on sharing resources 167
  - authorized facilities 29
  - defined to VTAM 9,25
  - definition of 9
  - displaying status of 54
  - starting and stopping 57
  - telecommunication security 169
- application-program indicators (see also specific indicators "clear," "start-data-traffic," etc.) 195
- application-program logon 49
- application programmer needs 20
- application programs, writing 67
- APPLID field of ACB 29
- appropriate available terminals for the network solicitor 63
- areas to consider when planning for VTAM 19
- ARG field of RPL 121
- asynchronous-flow
  - compared with synchronous-flow 100
  - indicators 195-198
  - messages 100
- asynchronous-flow indicators
  - example in Sample Program 2 130
  - summary 195-198
- asynchronous request handling
  - concept 75
  - example in Sample Program 2 127
- ATCCONxx 30
- ATCCON00 30
- ATCSTRxx 30
- ATCSTR00 30
- attention identifier (AID) 136
- AT&T Teletype TWX terminal 186
- AT&T 83B3 Selective Calling Stations 185
- audit-trail 145,147
- authorization-exit routine
  - affects on sharing resources 168
  - coding and installing 40
  - considerations 40
  - effects on sharing resources 168
  - output 40
  - telecommunication security 171
- authorized facilities, specified on APPL statements 29
- automatic dialing 161
- automatic logon
  - altered by the network operator 62
  - description of 48
  - specified in a definition statement 31
- auxiliary storage used by VTAM 16
- availability of VTAM facilities iv
  
- basic-mode communication 105
- BB (begin-bracket) indicator 195-198
- begin-bracket indicator 195-198
- Binary Synchronous Communications (BSC) terminals
  - communicating with 40
  - identifying 158
  - supported 186
- bind control signal 128
- block processing
  - authorizing the use of 29
  - effect on resource sharing 167
- bracket communication 95
  - begin-bracket indicator 95,195-198
  - bid indicator 95,195-198
  - enforcement 96
  - how it works 95
  - ready-to-receive (RTR) indicator 95
- BSC (see "Binary Synchronous Communications terminals")
- BTAM compared with VTAM 72
- buffering 46
  
- call-in terminals 160
- call-in/call-out terminals 161
- call-out terminals 161
- cancel indicator 92,195-198
- cataloged procedures 156
- chain
  - unit of work represented 92
  - use of chaining 93,133
- CHAIN field of RPL 136



- chaining
  - concept 92
  - example in Sample Program 2 129,133
  - 3601 output 129,133
- change-direction communication
  - command indicator 95,195-198
  - enforcement 95
  - how it works 95
  - request indicator 95,195-198
- CHANGE macro instruction 116
- changing line assignments 162
- changing line-scheduling parameters iv
- channel adapter support 154
- characteristics of logons and logoffs 47,50
- chase indicator 86,195-198
- CHECK macro instruction
  - description 112
  - examples in Sample Program 2 128,129,133
- CID (see "communications identifier")
- clear indicator 96,195-198
- CLOSE macro instruction
  - description 110
  - example in Sample Program 1 122
- closedown procedure 53
- closing VTAM down 53
- CLSDST macro instruction
  - description 111
  - PASS option 29,49,84
  - RELEASE option 84
  - use in disconnecting 114
- cluster control unit
  - defining 43
  - definition 43
  - minor node 43
  - starting and stopping traces for 63
- CLUSTER statement
  - contents 158,159
  - specifying
    - automatic logon 31
    - interpret table 32
  - used to define a cluster control unit 43
- CMD parameter in CHNGDIR operand 195-198
- coding and including installation exit routines
  - accounting 40
  - authorization 39
  - logon-interpret 41
- coexisting with other access methods 176
- Communicating Magnetic Card SELECTRIC ® Typewriter 185
- communicating with logical units
  - facilities 86
  - language 114
- communication between application programs and terminals 86
- communication macro instructions, record-mode
  - RECEIVE 111
  - RESETSR 111
  - SEND 111
  - SESSIONC 111
- communication, VTAM compared with BTAM 86
- communications controller
  - as a remote station 191
  - features not supported by VTAM 154
  - local and remote 153
  - models not supported by VTAM 154
  - models supported by VTAM 154
  - requirements for 153
- communications identifier (CID) 114
- COMP statement
  - contents 158
  - specifying
    - automatic logon 31
    - interpret table 32
  - used to define a terminal component 43
- comparing types of logons 49
- comparing VTAM application programs
  - with BTAM 72
  - with TCAM 72
- compatibility with TCAM 177
- completing a logon 50
- component, terminal
  - defining 43
  - minor node 43
- configuration data sets for TOLTEP 163-164
- connecting
  - example in sample programs 120
  - to application programs 70
  - to logical units/terminals 114
  - to nonexistent terminals 61
- connection 6,81
- connection and terminal sharing 168
- connection macro instructions
  - CLOSE 110
  - CLSDST 111
  - OPEN 110
  - OPNDST 110
- considerations for network-operator control 65
- continue-any mode
  - concept 102
  - example in sample program 121
- continue-specific mode
  - concept 102
  - examples in sample programs 121,127
- control-block macro instructions 111
- control blocks, application program
  - ACB 111
  - EXLST 111
  - handling in general 115
  - NIB 111
  - RPL 111
- CONTROL field of RPL 140
- controlling
  - a VTAM system 20,51
  - access to VTAM 175
  - buffering 47
  - connections 171
  - logon requests 172
- counter overflow 145
- CPU instructions required by VTAM 153
- CPU support 153
- creating the system 20,23
- CS parameter, example in sample program 120
- data flow through a VTAM system 12
- data sets
  - allocated to VTAM 18
  - for VTAM under DOS;VS 163-164
  - for VTAM under OS;VS 165,166
- DATAMGT macro instruction 25
- deactivating nodes, two modes of (see also "activating and deactivating") 56
- declarative macro instructions 111
- defining
  - application programs 29
  - automatic logons 31
  - control blocks, different methods 115
  - locally attached 3270s 26
  - logical units 159
  - logons and logoffs 31,50
  - major nodes 42
  - minor nodes 42-44
  - NCPs 25
  - nodes 42-44
  - terminal-operator logons 31
  - terminals 26,158
  - the network 25
  - VTAM start options 30
- device characteristics 116
- device support 181,183
- device transmission limit 64
- devices, allocating to VTAM 18
- DFASY exit-routine
  - concept 100

- example in sample program 139
- DIAGFILE 165
- direct-control access method 2
- disconnecting application programs and logical units/terminals 114
- disconnection
  - passing a logical unit 84
  - releasing a logical unit 84
- disk I/O, example of relationship to asynchronous processing of terminal inquiries 129
- DISPLAY command 54
- distinguishing between types of terminals 116
- distribution of function 10
- DO macro instruction 117
- DOS/VS
  - coexistence 178
  - CPU support 153
  - requirements
    - data sets 165-166
    - hardware 153
    - software 155
    - system generation 25,155
- DSECT instruction 112
- DSS (see "Dynamic Support System")
- dump utility for the NCP 144
- dumps
  - NCP, used by VTAM 147
  - OS/VS, used by VTAM 146
- dynamic configuration 5
- Dynamic Support System 156

- ECB operand 116
- efficient use of VTAM 166
- emulation mode
  - description 82
  - impact of VTAM on 161
  - not supported by VTAM 9
- end nodes 166
- end-bracket (EB) indicator 195-198
- end-of-day records 145
- ENDINTAB macro instruction 35
- enforcement of
  - bracket communication 96
  - change-direction communication 95
  - quiesce communication 95
- error
  - detection and feedback 149
  - recording 144
  - records 144,145
  - return codes 79
- error recovery 150,78
- establishing
  - procedures for using the system 20
  - standard logon messages 35
- event control block (ECB)
  - advantages compared with RPL exit-routine 76
  - general concept 76
- EX parameter of RPL RESPOND field, example in sample program 44
- exception message (compare with "exception response") 87
- exception response (compare with "exception message") 87
- EXECRPL macro instruction 112
- EXIT field of RPL 116
- exit-routines, application program
  - difference between installation exit-routines 78
  - examples in sample program 126
  - general concept 77
  - reenterability requirements 78
- EXLST control block
  - description 111
  - relationship to executable macros and other control blocks 113,114
- Extended Binary Coded-Decimal Interchange Code 183

- features of local 3270 defined in LOCAL statement 27
- filing
  - definitions
    - of application programs 29
    - of local 3270s 26
    - of NCPs 26
    - start options 30
- FME parameter in RPL RESPOND field, example in sample program 122
- FME response
  - exception FME responses 89
  - normal FME response 89
- forming a connection 81
- forms of manipulative macros 112
- four views of a VTAM system 16
- functions of VTAM 5

- GENCB macro instruction 112
- generating
  - a logon message 49-50
  - and tailoring VTAM 23
  - NCP support 25
  - support for devices at system generation 25
  - VTAM 25
- group
  - defining 43
  - minor node 43
- GROUP statement
  - contents 158,159
  - specifying
    - automatic logon 31
    - interpret table 32
    - used to define a group 43

- HALT command 53
- halting VTAM 53
- hardware-error recording 144
- hardware requirements for VTAM 153
- identification verification 158,170
- identifying problems in a VTAM system 144
- IDLIST statement 158,170
- implications of active
  - application program 30
  - terminal 31
- implicit allocation 18
- indicators (see also specific indicators "clear," "start-data-traffic," etc.) 195-198
- INTEST data set 163,165
- initial status of nodes
  - effect on sharing resources 168
  - specifying
    - major nodes 30
    - minor nodes
      - local 27
      - remote 158,159
- initiate-session 127
- initiating connection requests
  - as an authorized facility 29
  - from the application program 83
  - from the network-operator console 62
- initiation of VTAM 5,52
- INNODE statement 43
- input/output routine
  - example of 3270 input/output routine 136
  - example of 3600 input/output routine 131
- INQUIRE macro instruction 113
- installing a VTAM system 19
- INTAB macro instruction 34
- interface to TCAM, effects on accounting routine 41
- interpret tables
  - as part of terminal-operator logon

- for logical units 34
- for start-stop and BSC terminals 32
- creating 34
- specifying in definition:statements
  - for local 3270s 27
  - for remote terminals 158,159
- interrupting printout 93
- interruption request block (IRB) 78
- introduction to VTAM planning 19
- INTRPRET macro instruction
  - description 113
  - example of use 34
- I/O (input/output) areas 115
- I/O processing 5
- I/O routine (see "input/output routine")

KEEP option (of PROC field in NIB) 104

LBUILD statement 26

LDO control block 116

length of received input 121

LERAD exit-routine
 

- addressability 126
- example in Sample Program 2 129,130
- general concept 79
- processing 80

levels of control in VTAM 51

line (see "telecommunication line")

line-scheduling specifications 64

LINE statement
 

- contents 158,159
- specifying
  - automatic logon 31
  - interpret table 32
- used to define a line 43

loading NCPs 58,162

LOCAL statement
 

- contents 27
- specifying
  - automatic logon 27,31
  - interpret table 27
- used to define a local 3270 26

locally attached 3270s
 

- defined to VTAM 26
- supported by VTAM 181

LOGCHAR macro instruction 34

logical error 79

logical unit
 

- communicating with 86
- connecting 67,81,127
- description 13,83
- identifying, example 121
- relationship to application program 69
- terminal-initiated logon 32

logical-unit-status (LUS) indicator 195-198

logoffs 50,133

logon-interpret routines
 

- as installation exits 41
- input 35
- output 35
- specified on LOGCHAR macro instruction 34

logon-monitor facility (see "network solicitor")

LOGON operand of EXLST macro instruction 120

LOGON, part of standard logon message 35

logon request
 

- accepting 82
- queuing 82

logons
 

- as a means of increasing terminal sharing 167
- defining 31
- example in Sample Program 2 127
- restricting 172
- types of 47

LOSTERM exit-routine 125,130

LUS indicator 195-198

machine requirements for VTAM 153

macro instructions, VTAM application program, brief descriptions 69,110
 

- ACB 111
- CHANGE 116
- CHECK 112
- CLOSE 110
- CLSDST 111
- DO 116
- EXECRPL 112
- EXLST 111
- GENCB 112
- INQUIRE 113
- INTRPRET 113
- LDO 116
- MODCB 112
- NIB 111
- OPEN 110
- OPNDST 110
- READ 106
- RECEIVE 111
- RESET 106
- RESETSR 111
- RPL 111
- SEND 111
- SESSIONC 111
- SETLOGON 113
- SHOWCB 112
- SIMLOGON 113
- SOLICIT 106
- TESTCB 112
- WRITE 106

major nodes
 

- a definition 42
- that can be activated or deactivated 55

managing resources
 

- through application programs
- through NCP generation 168
- through VTAM definition 167

manipulative macro instructions 112

manual dialing 161

manual two-channel switch for the 3705 154

message length 121

messages
 

- concept 86
- length 86
- relationship to responses 86
- what they contain 86

methods of communication 94
 

- bracket communication 95
- change-direction communication 95
- quiesce communication 95

minor nodes
 

- a definition 42
- that can be activated or deactivated 56

MODCB macro instruction
 

- description 112
- example in Sample Program 2 127

MODIFY command, used
 

- to change line-scheduling specifications 64
- to start and stop the network solicitor 63
- to start and stop traces 63
- to start the dump utility program 64
- to start the trace-print utility 63
- to start TOLTEP 64

modifying
 

- NCPs 26
- the network solicitor 63

monitoring VTAM status 54

monopolizing VTAM resources 169

multiple console support (MCS) 156

- multiple definitions of the same node 29
- multiple-terminal access (MTA)
  - and the VTERM statement 158
  - defining to VTAM 158
- multipoint lines, and resource sharing 174
- multitasking support, required by VTAM 155
  
- name of application program
  - in automatic logon 31
  - in interpret table 34
  - in standard logon message 35
  - on APPL statement 29
- naming nodes, controlling 45,175
- NCP (see "network control program")
- NCP session limit 64
- NCPDUMP 165
- negative poll response limit 64
- NETSOL macro instruction 37
- NETSOL, name
  - of the network solicitor 38
  - specified in automatic logon for terminal-operator logon 38
- network control mode 9,156
- network control program
  - activating and deactivating 55
  - as a major node 43
  - defining to VTAM 25,157
  - displaying status of 55
  - dump data sets 163,165
  - functions required by VTAM 156
  - initial test 149
  - introduction 156
  - libraries 163,165
  - requirements 156
- network definition 25
- network operator
  - activities 51,20
  - commands
    - as a means of controlling the system 51
    - description of 51-64
  - considerations for 65
  - problem determination 143
  - prompting 52,53
  - use of system console 18
- network-operator logon
  - change of automatic logon 49
  - description 49
  - use of VARY command with 62
- network solicitor
  - acquire notification 38,37
  - as a start option 30
  - description of 36
  - effects on
    - accounting-exit routine 41
    - authorization-exit routine 40
  - messages 38
  - modifying 37
  - monitoring terminals 36
  - NETSOL
    - macro instruction 37
    - name (see "NETSOL, name")
  - RELREQ exit 38
  - use of interpret tables 34
  - validating logon messages 37
- NEX parameter of RPL RESPOND field, example in sample program 122
- NFME parameter of RPL RESPOND field, example in sample program 122
- NIB control block
  - description 111
  - relationship to executable macros and other control blocks 114
  - use in connecting 114
  
- node
  - addressable point 6
  - definition of 42
  - defined to VTAM 25
  - structure
    - for application programs 42
    - for local 3270s 42
    - for NCPs 42,44
    - symbolic names of 45
    - using definition statements to define 42-43
  - normal (positive) response 88
  
- OLT data sets for TOLTEP 163,165
- open ACB identifying application program 9
- OPEN macro instruction
  - description 110
  - general use 81
- opening a VTAM application program
  - compared with BTAM 81
  - description 113
  - example in sample program 118
  - operating system requirements 155
- OPNDST macro instruction
  - ACQUIRE option authorization 29
  - description 110
  - example in sample program 120
  - relationship to RPL and NIB 114
  - use in connecting 114
- OPTCD field of RPL 116
- orderly shutdown 53
- OS/VS
  - coexistence 178
  - CPU support 153
  - requirements
    - data sets 163
    - hardware 153
    - software 156
    - system generation 25,156
    - threshold specifications 46
- OS/VS2, and multiprocessing 154
- overlapping of VTAM, requests with other processing 74
- overlength input data 104
  
- Partitioned Emulation Programming (PEP) extension 1,161
- PASS option of CLSDST macro instruction, as an authorized facility 29
- passing connections
  - effects on resource sharing 168
  - an authorized facility 29
- passwords
  - and telecommunication security
    - for application-program authorization 175
    - for logon messages 173
  - on APPL statement 29
  - optional in logon message 34,35
- path 6
- PCCU statement 157
- PEP (see "Partitioned Emulation Programming (PEO) extension")
- permanent hardware errors 145
- physical error 79
- point-to-point lines, and sharing resources 169
- polling 101
- polling delays, changing 64
- port
  - concept of 160
  - defined by 160
  - displaying status of 54
  - minor node 43
  - name of 43
  - port solicitor, effect on accounting-exit routine 41

POST operand of RPL 115  
 posting an ECB  
   in an RPL exit-routine 128  
   in VTAM 76  
 primary purpose of VTAM 1,2  
 priority of messages 100  
 priority requests for storage 150  
 procedures for the network operator 20  
 processing software errors 167  
 program information block (PIB) 78  
 programming alternatives 140  
 prompting during start processing 52  
 protecting sensitive data 176  
 purpose of  
   node structure 44  
   VTAM 1,2

QC indicator (see "quiesce-completed indicator")  
 QEC indicator (see "quiesce-at-end-of-chain indicator")  
 queuing correction requests 84  
 queuing for logon 82  
 quick closedown 53  
 quiesce  
   communication 95  
   enforcement 95  
   how it works 95  
 quiesce-at-end-of-chain (QEC) indicator  
   concept 93  
   example in Sample Program 2 130  
   summary 195-198  
 quiesce-completed (QC) indicator 93,195-198  
 quiescing  
   concept 93  
   example in Sample Program 2 139  
   example of use 93,139

RAS (also see "reliability, availability, serviceability")  
 RAS libraries 19  
 READ macro instruction 117  
 read-only RPL, example in sample programs 123,137  
 ready-to-receive indicator 195-198  
 RECEIVE macro instruction  
   description 111  
   examples in sample programs 120,127  
 receiving input, example in sample programs 121  
 RECLen field of RPL 121  
 recording hardware errors 144  
 recording software errors 145  
 record-mode (SEND/RECEIVE mode) 130  
 redefining NCPs 26  
 reenterability  
   exit-routines except for LERAD and SYNAD 126  
   LERAD and SYNAD 80  
   parameter list forms for programs that must be  
   reenterable 112  
 reestablishing connections 151  
 release-quiesce (RELQ) indicator  
   description 83,195-198  
   example in Sample Program 2 130  
 reliability and availability support in VTAM 148,143  
 RELQ indicator (see "release-quiesce indicator")  
 RELREQ exit-routine 84  
 RELREQ exit in the network solicitor 38  
 remote communications controller, a definition 191  
 remote NCP  
   defining 25  
   node structure 44  
 remote station 191  
 replacing the network solicitor 39  
 REQ parameter of CHNGDIR operand 195-198  
 request-recovery indicator 195-198  
 request shutdown (RSHUTD) indicator 195-198  
 required CPU instructions 153

requirements  
   for communications controllers 153  
   for DOS/VS 155  
   for NCPs 156  
   for OS/VS 168  
   operating system 155  
   reenterability 112  
   system 3  
 RESET macro instruction 117  
 RESETSR macro instruction  
   description 111  
   example in Sample Program 1 123  
 resources that can be shared 166  
 RESP exit-routine  
   concept 100  
   example in Sample Program 1 123  
   example in Sample Program 2 130  
 RESPOND field of RPL 121  
   example in sample programs 122,131  
 responded output  
   concept 87,89  
   how specified 115  
 responses  
   as acknowledgments 86  
   concept 86  
   relationship to messages 86,87  
   what they contain 86  
 restrictions in communicating with 3270 in record-mode 105  
 RPL control block  
   description 111  
   relationship to executable macros 113-115  
 RPL exit-routine  
   advantages 76  
   concept 76  
   example in Sample Program 2 128  
 RQR indicator (see "request-recovery indicator")  
 RRN response  
   concept 89  
   exception RRN response 89  
   normal RRN response 89  
 RSHUTD indicator (see "request-shutdown indicator")  
 RTYPE operand of RECEIVE macro, example in sample  
 program 120

sample logic of a DFASY exit-routine 139  
 sample logic of a LOGON exit-routine 127  
 sample logic of a RESP exit-routine 137  
 sample programs 117  
   Sample Program 1 (synchronous) 118  
   Sample Program 2 (asynchronous) 123  
 scheduled output  
   concept 86,88  
   how specified 115  
 SCIP exit-routine 100  
 SDLC (see "synchronous data link control")  
 SDLC cluster controller 6  
 SDT indicator (see "start-data-traffic indicator")  
 security considerations for standard logon 174  
 security, telecommunications 169  
 SEND main instruction  
   description 111  
   example of sending a message in a sample program 122  
   example of sending a response in a sample program 121-122  
 SEQNO field of RPL 134  
 sequence number 92,137  
 sequence number recovery 96,137  
 sequencing 92  
 services performed by VTAM 1  
 SESSIONC indicators 195-198  
 SESSIONC macro instruction  
   description 111  
   use in sequence number recovery 96,137

set-and-test-sequence number (STSN) indicator 97,137,195-198  
 SETLOGON macro instruction  
   description 113  
   example of use in sample program 120  
 sharing  
   and managing resources 166  
   logical units among programs 83  
   parameter lists 112  
   resources 13,166  
 SHOWCB macro instruction  
   description 112  
   example of use in sample program 134  
 SHUTC indicator (see "shutdown-completed indicator")  
 SHUTD indicator (see "shutdown indicator")  
 shutdown indicator 195-198  
 shutdown-completed indicator 195-198  
 signal indicator 195-198  
 significant elements in a VTAM system 16  
 SIMLOGON macro instruction  
   an authorized facility 29  
   description 83  
   used to initiate application-program logon 49,113  
 simulated logon requests 83  
 SNAP macro instruction 147  
 SOLICIT macro instruction 117  
 specific-mode  
   compared with any-mode 101  
   concept 101  
 specifying  
   automatic logons 31  
   buffers 46  
   interpret tables 34  
   terminal-operator logons 31  
 standard logon  
   and telecommunication security 174  
   establishing 35  
   for logical units 36  
   for start-stop and BSC terminals 35  
 START command 52  
 start-data-traffic (SDT) indicator  
   description 96,195-198  
   example in Sample Program 2 127  
 start options  
   from the console 52  
   predefined 30  
 start-stop terminals  
   communicating with 116  
   support 184  
 start-time, and VTAM buffers 46  
 starting and stopping  
   application programs 57  
   VTAM 52,53  
   VTAM facilities 62  
 storage interlock 150  
 storage management 149  
 storage, obtaining for RPL and other areas 126,127  
 storage pools 46,149  
 STSN (see "set-and-test-sequence number indicator")  
 summary  
   of VTAM application program indicators 195-198  
   of VTAM application program macro instructions 110  
 support  
   for local 3270s 181  
   for remote terminals 183  
   for switched networks 159  
 support macro instructions  
   CHECK 112  
   EXECPRL 112  
   INQUIRE 113  
   INTRPRET 113  
   SETLOGON 113  
   SIMLOGON 113  
 SUPVR macro instruction 25  
 symbolic names, and telecommunication security 175  
 SYNAD exit-routine  
   addressability 126  
   example in Sample Program 2 125,130,137  
   general concept 79  
   processing 81  
 synchronous data link control, devices 183  
 synchronous-flow  
   compared with asynchronous-flow 100  
   indicators 195-198  
   messages 100  
 synchronous request handling  
   examples in Sample Program 1 120,122  
   general concept 76  
 SYSSLB 166  
 system console 18  
 system generation  
   DOS/VS 25,155  
   OS/VS 25,156  
 system support 3  
 System/3 188  
 System/370 189  
 System/7 185,188  
 SYS1.DUMP 163-164  
 SYS1.LINKLIB 163-164  
 SYS1.LOGREC 163-164  
 SYS1.LPALIB 163-164  
 SYS1.MACLIB 163-164  
 SYS1.NUCLEUS 163-164  
 SYS1.PARMLIB 163-164  
 SYS1.PROCLIB 163-164  
 SYS1.SVCLIB 163-164  
 SYS1.TRACE 163-164  
 SYS1.VTAMLIB 163-164  
 SYS1.VTAMLST 163-164  
 SYS1.VTAMOBJ 163-164  
 tailoring VTAM 29  
 TCAM 72  
 telecommunication lines  
   changing assignments of 162  
   defined to VTAM 43  
   displaying status of 55  
   minor node 43  
   starting and stopping traces for 63  
 telecommunication network 6  
 telecommunication security  
   points of control 169  
   through VTAM 169  
   ways to control 169  
 telecommunication system 6  
 teleprocessing 5  
 Teleprocessing Online Terminal Test Executive Program (TOLTEP)  
   data sets 163,165  
   description 147  
   effect on accounting-exit routine 41  
   starting 64  
   teleprocessing subsystem 10,13  
 TERMINAL statement  
   contents 158  
   relation to PCCU statement 43  
   specifying  
     automatic logon 31  
     interpret table 32  
   used to define  
     a port to VTAM 43  
     a remote communications controller 43  
     a terminal to VTAM 43  
 terminal (see also "logical unit")  
   activating and deactivating 59,61  
   as the boundary of the network 11  
   connecting to 81  
   defined to VTAM  
     local 26  
     remote 158,159

- definition of 10
- displaying status of 54
- minor node 43
- queuing 19
- starting and stopping traces for 49
- terminal-operator logon
  - defining 31
  - description 48
- termination of VTAM 5,53
- TESTCB macro instruction 112
- thresholds, for buffers 46,150
- TOLTEP (see "Teleprocessing Online Test Executive Program")
- TPEND exit 53,122
- trace-print utility program
  - description 146
  - starting 64
- traces
  - generalized trace facility (GTF) 146
  - operating systems 146
  - NCP 146
  - VTAM's 146
- transmitting data 1
- TRFILE 166
- TRUNC option 104
- tuning a program 140
- TWX terminals, identifying 170
- types of logons 47

- upward compatibility 155
- USENSEO field of RPL 133
- user (USER) field of RPL
  - example in Sample Program 2 128
  - relationship to USERFLD of NIB 115
- USERFLD of NIB, relationship to USER field of RPL 115
- using
  - NCP configuration restart 151
  - same deck to generate and to define an NCP 26
  - VTAM node structure 44
  - VTAM RAS facilities 143
- UTERM parameter of TERMINAL statement 181

- validating logon messages in the network solicitor 34
- VARY command, used to
  - activate and deactivate nodes 55
  - activate TOLTEP 64
  - initiate the network-operator logon 62
- verifying terminal identifications 170
- Virtual Telecommunications Access Method (VTAM)
  - application program
    - concepts and facilities 67
    - macro instructions (see "macro instructions" for individual macro references) 110
    - overview 67
    - processing part 67
    - relationship to BTAM application program 72
    - relationship to a logical unit 67
    - relationship to TCAM application program 72
    - relationship to VTAM system 67
    - telecommunications part 68
  - as a systems manager 5,6
  - buffering 46
  - commands 51
  - connecting terminals 19
  - data sets
    - under DOS/VS 165,166
    - under OS/VS 163
  - definition 19,23
  - device support 181,183
  - dynamic configuration 76
  - example of a system
    - physical configuration 16
    - viewed by
      - application programs 18
      - operating system 16
      - VTAM 16-17
  - features 2-3
  - functions 5
  - in operation 19
  - interfaces to 1
  - processing 5
  - libraries 16
  - logons 47
  - network control 19,20
  - node structure 41
  - RAS facilities 143,144
  - role in a telecommunication system 5
  - sharing resources 166
  - slowdown 150
  - support for
    - communications controllers 153
    - CPUs 153
    - terminals 181,183
  - telecommunication security 169
  - terminal 10
  - traces
    - nodes that can be traced 63
    - starting and stopping the trace 63
    - starting the trace-print utility 64
  - transmitting data 1
- VIDLIST statement
  - contents 158
  - used to identify terminals 170
- VSAM compared with VTAM 110
- VTAM (see "Virtual Telecommunications Access Method")
- VTERM statement
  - contents 158
  - specifying
    - automatic logon 31
    - interpret table 32
  - used to identify MTA terminals 158

- wait routine 125
- work areas 115,126
- work station 69
- World Trade Telegraph Terminals 185
- Western Union (WU) Plan 115A 185
- WRITE macro instruction 117

- 1050 Data Communication System 184
- 2740 Communication Terminal, Model 1 184
- 2740 Communication Terminal, Model 2 185
- 2741 Communication Terminal 185
- 2770 Data Communication System 186
- 2780 Data Transmission Terminal 187
- 2980 General Banking Terminal System 187
- 3270 attention identifier 136
- 3270 command character 136
- 3270 communication
  - example of input/output routine 136
  - in record-mode 105,136
  - restrictions in record-mode 105
  - sample program 123
- 3270, differences between logical unit 136
- 3270 Information Display System
  - locally attached
    - features supported 181
    - defined to VTAM 26
  - remotely attached 187

3600 Finance Communication System 183  
3601 Finance Communication Controller  
    example of chaining output to a 3601 logical unit 133  
    example of input/output routine 131  
    sample program 123  
3650 Retail Store System 183  
3660 Retail Store System 183  
3704, 3705 (see "communications controllers")  
3735 Programmable Buffered Terminal 188  
3740 Data Entry System 188  
3780 Data Communications Terminal 188  
3790 Communications System 183





GC27-6998-0

*Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

How did you use this publication?

- |  |   |
|--|---|
| <input type="checkbox"/> As an introduction                  | <input type="checkbox"/> As a text (student)    |
| <input type="checkbox"/> As a reference manual               | <input type="checkbox"/> As a text (instructor) |
| <input type="checkbox"/> For another purpose (explain) _____ |   |

Please comment on the general usefulness of the book; suggest additions, deletions, and clarifications; list specific errors and omissions (give page numbers):

Cut or Fold Along Line

What is your occupation? \_\_\_\_\_

Number of latest Technical Newsletter (if any) concerning this publication: \_\_\_\_\_

Please include your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)



GC27-6998-0

*Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

How did you use this publication?

- |  |   |
|--|---|
| <input type="checkbox"/> As an introduction                  | <input type="checkbox"/> As a text (student)    |
| <input type="checkbox"/> As a reference manual               | <input type="checkbox"/> As a text (instructor) |
| <input type="checkbox"/> For another purpose (explain) _____ |   |

Please comment on the general usefulness of the book; suggest additions, deletions, and clarifications; list specific errors and omissions (give page numbers):

Cut or Fold Along Line

What is your occupation? \_\_\_\_\_

Number of latest Technical Newsletter (if any) concerning this publication: \_\_\_\_\_

Please include your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Cut or Fold Along Line

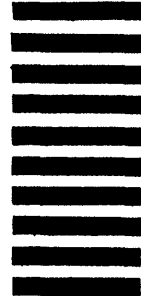
**Your comments, please . . .**

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold

Fold

First Class  
Permit 40  
Armonk  
New York



**Business Reply Mail**  
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

**International Business Machines Corporation  
Department 63T  
Neighborhood Road  
Kingston, New York 12401**

Fold

Fold



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

**IBM World Trade Corporation  
821 United Nations Plaza, New York, New York 10017  
(International)**

**IBM**

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

**IBM World Trade Corporation  
821 United Nations Plaza, New York, New York 10017  
(International)**