GC27-6995-5
File No. S370-30

Systems

VTAM Macro Language Reference

Virtual Telecommunications Access Method (VTAM)

VTAM Level 2

DOS/VS
OS/VS1
OS/VS2 SVS
OS/VS2 MVS

IBM

Sixth Edition (April 1978)

This edition is a major revision of, and supersedes, the previous edition, GC27-6995-4, and its Technical Newsletters, GN27-1575, GN27-1581, and GN31-0659. The changes for this edition are described in the Summary of Amendments dated April 1978.

This edition corresponds to VTAM Level 2 in:

DOS/VS Releases 32, 33, and 34

OS/VS1 Releases 5 and 6

Component release to OS/VS2 SVS Release 1.7

VTAM 2 component release to OS/VS2 MVS Release 3.7

Changes are continually made to the information in IBM system publications. Before using this publication in connection with the operation of IBM systems, consult the *IBM System/370 Bibliography*, GC20-0001, to find out which editions are applicable and current.

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

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

Summary of Amendments (April 30, 1978) to GC27-6995-4 by Revision GC27-6995-5

VTAM Level 2 for DOS/VS, OS/VS1, OS/VS2 SVS, and OS/VS2 MVS

Changed Documentation

Additional return codes for the CLOSE macro instruction are given (OS/VS).

A flowchart for the ATTN exit routine has been added, and the DFASY and RESP exit routine flowcharts have been clarified.

The RPL CODESEL field is clarified.

RPL RTNCD-FDBK2 codes have been clarified and expanded.

Many minor technical and editorial changes have been made throughout this book.

PREFACE

This book is a reference manual that contains detailed information on the macro instructions used with the Virtual Telecommunications Access Method (VTAM). The macro instructions are used to write the telecommunication portions of application programs that communicate with terminals through VTAM. This manual provides the specifications needed to code such programs.

Although a program can be coded directly from this manual, those who are unfamiliar with VTAM application programming should first read *VTAM Macro Language Guide,* GC27-6994 and, optionally, *Supplement to the VTAM Macro Language Guide for the Program Operator,* GC27-0036. The *Guide* and its supplement form companion books to this one.

The beginning of this book lists the services provided by VTAM and indicates the macro instructions that are used to request each service. The beginning of the book also explains the conventions used throughout the book to indicate how the macro instructions are to be coded.

The rest of the book (except for the appendixes) contains detailed descriptions of the macro instructions, arranged in alphabetic order. Each description is presented in a fixed format with the information about each macro instruction presented in the same sequence.

With few exceptions, VTAM macro instructions can be coded without regard for the particular operating sytem (DOS/VS, OS/VS1, OS/VS2 SVS or OS/VS2 MVS) under which the program will be running. When there is an exception to this, that exception is identified in the macro instruction description.

Appendix A is a summary of the control block fields you use with each macro instruction. Once you have become familiar with the macro instructions, you will be able to use this appendix as a quick reference source.

Appendix B indicates the communication control characters that VTAM inserts into outgoing data and recognizes in incoming data. This information is shown for each BSC and start-stop device supported by VTAM.

Appendixes C and D describe the return codes that are passed to the application program upon completion of each VTAM macro instruction.

Appendixes E and F describe the operand formats and special forms of the GENCB, MODCB, SHOWCB, and TESTCB macro instructions.

Appendix G summarizes the contents of the general purpose registers upon completion of VTAM macro instructions.

Appendix H shows the format of the application program control blocks and the DSECTs needed to access these control blocks with assembler instructions.

Appendix I contains information relating to specific devices and the way the VTAM macro instructions should be used with the devices.

Appendix J shows the format of the bind area and the DSECT needed to access the information in it.

The appendixes are followed by a glossary and an index. The index includes page numbers for all of the macro instruction operands and all of the fixed values that can be supplied with the operands.

The reader should be familiar with *Introduction to VTAM,* GC27-6987, and with those parts of *OS/VS and DOS/VS Assembler Language,* GC33-4010, that explain the rules for coding assembler expressions. The reader should also be familiar with the characteristics of the devices with which the program will be communicating, with the line discipline (if start-stop, BSC, or SDLC) that will be used with each one, and with teleprocessing concepts in general. Those unfamiliar with teleprocessing concepts can read *Introduction to Data Communications Systems,* SR20-4461.

A few portions of the VTAM language cannot be fully utilized without a working knowledge of the network control program of the communications controller. If the reader is not familiar with this control program, a copy of the following publication should be obtained:

IBM 3704 and 3705 Control Program Generation and Utilities Guide and Reference Manual, GC30-3008. This publication is referred to as the *NCP Generation and Utilities Guide* in the remainder of this manual.

CONTENTS

FIGURES

FUNCTIONS PROVIDED BY THE VTAM MACRO INSTRUCTIONS

The Virtual Telecommunications Access Method (VTAM) provides a program running under a virtual storage operating system with the ability to do the following:

- Connect an application program with the terminals of a telecommunications network
- Communicate with those terminals
- Display and control the status of the network

The VTAM language described in this book is the set of macro instructions that are available to the user to request these functions.

VTAM provides four macro instructions to communicate with a logical unit and, optionally, a BSC or local non-SNA 3270. These macro instructions (SEND, RECEIVE, RESETSR, and SESSIONC) are designated as record-mode macro instructions. The macro instructions used to communicate with local non-SNA 3270, BSC and start-stop terminals (READ, WRITE, RESET, SOLICIT, DO, LDO, and CHANGE) are designated as basic-mode macro instructions. All other macro instructions described in this book can be used for both BSC and start-stop terminals and for logical units. The BSC and local non-SNA 3270 is a special case; it can be used in either basic mode or record mode.

It is a good programming practice to set control block fields in user-written application programs to zero after they are used if the control block is to be reused. Desired operands should be specified or reset to the system default. ACB, NIB, and RPL fields that are not used by a macro instruction should be set to zero unless otherwise indicated.

The program using VTAM can request that VTAM perform or initiate the following actions; the macro instruction used to request each one is shown in parentheses.

Record Mode

- Send a message or response to a logical unit (SEND).
- Receive a message or response from a logical unit (RECEIVE).
- Cancel a RECEIVE prematurely; switch a logical unit's continue-specific mode or continue-any mode (RESETSR).
- Send a Clear, Set and Test Sequence Number, or Start Data Traffic command to a logical unit and receive the reply for that command (SESSIONC).

Basic Mode

- Obtain data from one or a group of terminals and keep the data in VTAM buffers. Repeat this action until a specified amount of data has been received (SOLICIT).
- Using data already obtained from any terminal or from a specified terminal, move the data from VTAM buffers to an area in user storage (READ).
- Obtain data from a specific terminal and move it directly into user storage (READ).
- Transmit data from an area in program storage to a specified terminal (WRITE).
- Automatically follow an output operation with an input operation (WRITE).
- Cancel an I/O operation prematurely; reset an error lock set for a device (RESET).

- Initiate one or more I/O operations using logical device orders (DO).
- Perform I/O operations that are unique to the 3270 Information Display System, System/3, and System/370 that cannot be performed using READ and WRITE (LDO).

Common to Basic Mode and Record Mode

- Check the completion status of any of the above activities (CHECK).
- Execute any request defined by an RPL (EXECRPL).

The I/O and I/O-related facilities listed above can be used by a program only after certain preparation has taken place. Control blocks must be built that describe the specific nature of the I/O operation to be performed. Since VTAM allows terminals to be used first by one program and then by another, connection between the program and the terminal must be established before any I/O activity can take place. The connection itself needs control blocks that describe the specific nature of that operation.

The following VTAM services prepare for and support subsequent I/O activity.

- Create a control block that contains the parameters for a connection or I/O operation (RPL).
- Create a control block that identifies the program to VTAM and the telecommunications network (ACB).
- Create a control block containing entry points for routines to be entered when certain events occur, such as attention interruptions, hardware errors, or a terminal's request for connection to the program (EXLST).
- For each terminal, create a control block that contains information that affects subsequent communication with that terminal (NIB).
- Generate any of the above control blocks during program execution rather than during program assembly; optionally generate them in dynamically allocated storage (GENCB).
- Test, extract, or modify the parameters contained in these control blocks (TESTCB, SHOWCB, MODCB).
- Identify the program to VTAM and the telecommunication network (OPEN).
- Establish connection with one or more logical units (OPNDST).
- Simulate a terminal's request for connection, so that a user-written routine that handles such requests will be invoked (SIMLOGON).
- Allow logon requests to be directed at the application program, notify VTAM that the application program is no longer accepting logon requests, or indicate that the application program is once again accepting logon requests (SETLOGON).
- Obtain the device characteristics, session parameters, or logon message of a terminal requesting connection, or find out how many terminals are currently connected to the program and how many are waiting to become connected (INQUIRE).
- Disconnect a terminal from the application program; optionally request that the disconnected terminal be connected to another program (CLSDST).
- Disconnect the application program from VTAM and the telecommunications network (CLOSE).

The following VTAM services permit the application program to display and control the status of the network:

- Send a network operator command to VTAM (SENDCMD).
- Receive a VTAM message in reply to a network operator command or receive an unsolicited VTAM message (RCVCMD).

CATEGORIES OF VTAM MACRO INSTRUCTIONS

Throughout the macro instruction descriptions and the appendixes of this book, you will encounter the terms *manipulative*, *declarative*, *RPL-based*, and *ACB-based*, macro instructions. These terms refer to categories of VTAM macro instructions. Figure 1 shows these categories and identifies the macro instructions that are included in each one.

Declarative Macros

Macros	Description
ACB EXLST RPL NIB LDO (Basic Mode Only)	These build control blocks during program assembly. They are the only nonexecutable macro instructions.

Manipulative Macros

Macros	Description
GENCB MODCB SHOWCB TESTCB	These build and manipulate control blocks during program execution.

ACB-Based Macros

Macros	Description
OPEN CLOSE	These open and close the application program's ACB

RPL-Based Macros

These are used to request connection and data transfer. They all use an RPL and, with the exception of CHECK, permit RPL modifications to be specified in the macro instruction itself.

Connection Macros

- OPNDST
- CLSDST

I/O Macros

- Record Mode Only
  - SEND
  - RECEIVE
  - RESETSR
  - SESSIONC
- Basic Mode Only
  - SOLICIT
  - READ
  - WRITE
  - RESET
  - DO

Macros that Support Connection or I/O

- CHANGE (Basic Mode Only)
- CHECK
- EXECRPL
- INQUIRE
- INTRPRET
- SETLOGON
- SIMLOGON

Network Operator Macros

- SENDCMD
- RCVCMD

Figure 1. Categories of VTAM Macro Instructions

REGISTER RESTRICTIONS

Registers 2-12 (and only these registers) can be used with the macro instructions described in this book. This restriction applies both to registers used for the macro instruction itself (register notation for macro instruction operands) and to registers the programmer sets and expects to remain unmodified by the macro instruction.

The restriction to registers 2-12 applies regardless of the type of operating system, and regardless of whether the macro is an RPL-based, ACB-based, or manipulative macro instruction.

In addition, register 1 can be used to supply an RPL address for any RPL-based macro instruction. Example: SEND RPL=(1)

Before issuing an executable macro instruction, register 13 must contain the address of an 18-word save area. VTAM will not modify the contents of register 13.

There is no error return code that indicates an attempted misuse of registers; VTAM does not enforce the register restriction in any way. The results of using registers 0, 1, 14, or 15 (other than for the exception cited above) are unpredictable.

Readers interested in a description of how VTAM uses the restricted registers should refer to the table in Appendix G.

A NOTE ON VTAM-VSAM SIMILARITIES

The Virtual Storage Access Method (VSAM) is an access method for direct access storage devices (DASDs). Like VTAM, it is available to programs running under virtual storage operating systems. There is considerable similarity between the two access methods with regard to control block names and fields, control block manipulation, and general approach to request handling.

Both access methods use an ACB. The VTAM ACB essentially represents an application program. In VSAM, however, where the user has no need of an application program control block, the ACB represents the data set and is analogous to a DCB or DTF. Both types of ACBs are, however, objects of the OPEN macro instruction, and VSAM and VTAM ACBs can be opened with one macro instruction.

Both types of ACBs can contain pointers to an exit list. Both VSAM and VTAM exit lists contain addresses of routines to be entered when error conditions occur (LERAD and SYNAD exit routines) and addresses of routines to be entered when special situations occur.

Both access methods follow the same general I/O-request procedure: An I/O macro instruction is issued that indicates an RPL. The RPL in turn contains information about the request, such as the location of the I/O work area or whether the request is to be handled synchronously or asynchronously.

Finally, both access methods use the same macro instructions—GENCB, MODCB, TESTCB, and SHOWCB—to generate and manipulate their respective ACB, EXLST, and RPL control blocks.

Although the control blocks are similar in name, function, and (to some extent) content, the control blocks of one access method are not interchangeable with the corresponding control blocks of another.

To make control blocks unique, a special VTAM operand is used when the control block is generated. By specifying AM=VTAM on the ACB, EXLST, or RPL macro instruction, the control block is generated in VTAM-compatible form. Omitting this operand causes a VSAM-compatible control block to be built.

A NOTE ON CONTROL BLOCK MANIPULATION

The application program control blocks (ACB, EXLST, RPL, and NIB) can be examined and modified two ways during program execution: The application program can use the manipulative macro instructions (MODCB, TESTCB, or SHOWCB) or it can use IBM-supplied DSECTs.

The manipulative macro instructions are essentially branches to access method routines that perform the control block manipulations specified on the macro. Their advantage is their ease of use and the freedom from reassembly they provide should control block formats be changed in future releases of VTAM or should you change from one operating system to another. (To avoid reassembly, GENCB must be used in place of ACB, EXLST, RPL, and NIB declarative macros. MODCB, rather than RPL-based macros, must be used to modify all RPLs.)

The DSECTs provide labeled overlays for each of the control blocks for each operating system (the OS/VS1, OS/VS2 SVS, and OS/VS2 MVS ACB, EXLST, and RPL control blocks are identical, and the NIB is identical for all four operating systems). Their advantage is the improved performance (less system overhead) available through user-written assembler instructions. (Their disadvantage is that reassembly may be required for future releases of VTAM. The impact of reassembly can be lessened, however, by keeping the teleprocessing portions of the application program – that is, the VTAM macro instructions – separate from the processing portions.) The general use of DSECTs is described in "The DSECT Instruction" in *OS/VS and DOS/VS Assembler Language,* GC33-4010.

The manipulative macro instructions are described alphabetically in this manual; tabulated information about them is contained in Appendixes E and F. The formats and DSECTs for control blocks are described in Appendix H.

HOW THE MACRO INSTRUCTIONS ARE DESCRIBED

First, for an understanding of how macro instructions descriptions are arranged in this book, look at Figure 2. The balance of this section explains the conventions used in this figure.

Assembler Format Table

Each macro instruction description contains a three-column table that shows how the macro instruction is to be coded. Since macro instructions are coded in the same format as assembler instructions, the three columns correspond to an assembler instruction's name, operation, and operand fields. This table is subsequently referred to as the macro instruction's assembler format table.

Name: The macro instruction name provides a label for the macro instruction. The name, if used, can be specified as any symbolic name valid in the assembler language.

Operation: This field contains the mnemonic operation code of the macro instruction. It is always coded exactly as shown.

Operands: The operands provide information for the macro expansion program in the assembler. Generally, the information provided by the operands is made part of a parameter list provided to VTAM during program execution. All of the macro instruction's operands are indicated in the operands column of the assembler format table.

Types of Operands: All operands are either *keyword* or *positional* operands. Most of the VTAM macro instruction operands are keyword operands.

Keyword operands consist of a fixed character string (the operand keyword), an equal sign, and a single or multiple operand value. The presence of the equal sign distinguishes keyword from positional operands. Keyword operands do not have to be coded in the order shown in the operands column. For example, a macro having a LENGTH=*data length* operand and an AREA=*data area address* operand (as indicated in the operands column) could be coded as either

AREALEN=132,AREA=WORK
or
AREA=WORK,AREALEN=132

Keyword operands must be separated by commas. If a keyword operand is omitted, the commas which would have been included with it are also omitted.

There are a few instances in the VTAM macro instructions where more than one value can be coded after the keyword, but parentheses are required to do this. For example, an operand specified as

FIELDS= {field name | (field name,...) }

can be coded as

FIELDS=RECLEN
or
FIELDS=(RECLEN)

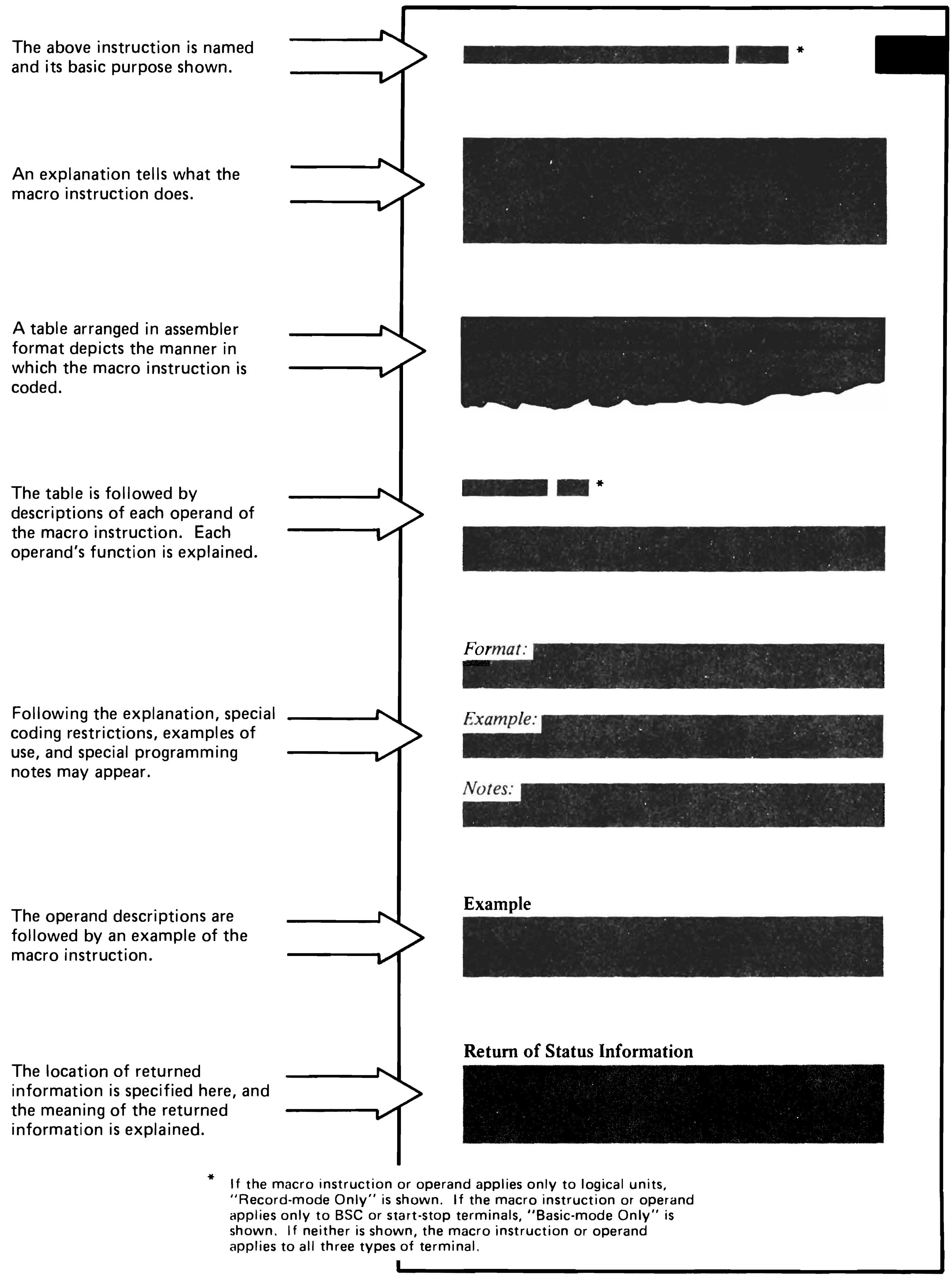


Figure 2. Basic Structure of the Descriptions of Each Macro Instruction

when only one field name is used. When more than one field name is used, however, the names must be enclosed in parentheses:

FIELDS=(RECLEN,RTNCD,FDBK2)

The field names must be separated by commas. If a field name is omitted, the comma that would have been included with it is also omitted. For example, omitting the first field name from the previous example would result in:

FIELDS=(RTNCD,FDBK2)

Positional operands (used in OPEN and CLOSE macros only) must be coded in the exact order shown in the operands column. Positional operands are separated by commas, as are all operands, but if a positional operand is omitted, the surrounding commas must still be entered. For example, consider a macro that has three positional operands DCB1, INOUT, and ACB1. If all three are used, they are coded as

DCB1,INOUT,ACB1

but if only DCB1 and ACB1 are wanted, they are coded as

DCB1,,ACB1

If the last positional operand or operands are omitted, the trailing comma or commas should not be coded.

Operand Notation: A notational scheme is followed in the operands column to show how, when, and where operands can be coded. The notational symbols are never coded.

- A vertical bar (|) means "exclusive or." For example, A | B means that either A or B (but not both) should be coded. Such alternatives can also be shown aligned vertically, as shown in the next paragraph.
- Braces ({ }) are used to group alternative operand values. One of the alternative values enclosed with the braces must be chosen. The alternatives may be shown vertically:

  $$\text{OPTCD=}\begin{Bmatrix}\text{COND}\\ \text{UNCOND}\\ \text{LOCK}\end{Bmatrix}$$

  or they can appear on one line:

  OPTCD= {COND | UNCOND | LOCK }

  Both expressions are equivalent. Note how the vertical bar is used to separate alternative values that appear on one line. When the grouping of alternatives on one line is unambiguous, the braces are usually omitted:

  OPTCD=COND | UNCOND | LOCK
- An underscored value means that if the operand is omitted, the macro will be expanded as though the underscored value had been coded. This alternative is called the assumed value, or default value. For example;

  OPTCD=<u>COND</u> | UNCOND | LOCK

  Here COND is the assumed value. If the OPTCD operand is omitted, OPTCD=COND is assumed by the assembler.
- Brackets ([ ]) denote optional operands. In the following example, the ERET operand is optional.

AM=VTAM
[,ERET=error routine address]

- An ellipsis (...) indicates that whatever precedes it (either an operand value or an entire operand) can be repeated any number of times. An operand appearing as

  PROC=(processing option,...)

  could, for example, be coded as:

  PROC=(CONFTXT,DFASYX,RESPX)

- Parentheses, equal-signs, and uppercase characters must be coded exactly as shown in the operands column. Lowercase words represent values that the user must supply.

Comments and Continuation Lines: Comments may contain any characters valid in the assembler language. Comments can be continued on more than one card by placing an asterisk in column 1 as shown in the example below. In this publication, the comments field is not shown in the macro's assembler format table.

Operands can also be continued on additional cards as shown below. Note that if the operands are not extended to column 71, they must be separated after a comma. The continuation character in column 72 can be any nonblank character, but it cannot be a character of an operand. Comments must by separated from operands by at least one blank. Throughout the rest of this publication, the continuation characters are not shown.

Name	*Operation*	*Operands*
LABEL1	**OP1**	**OPERAND1,OPERAND2,OPERAND3,OPERX AND4,OPERAND5 THIS IS ONE WAY**
LABEL2 *	**OP2**	**OPERAND1,OPERAND2, AND THIS X OPERAND3,OPERAND4, IS ANOTHER WAY**

column 1 *column 16* *column 72*

Operand Descriptions

Following the assembler format table, each operand is named and described. Every operand description begins with an explanation of the operand's function. If the operand has more than one fixed value that can be supplied with it, the operand description also explains the effect that each value has on the action performed by the macro instruction.

Operand Format: The operand description may include a description of the format in which the operand should be coded. This description is provided when the format is an exception to these general rules:

- When a *quantity* is indicated (for example, RECLEN=data length), you can specify the value with unframed decimal integers, an expression that is equated to such a value (for example, RECLEN=TEN, where, TEN EQU 5*2), or the number of the register (enclosed by parenthes) that will contain the value when the macro instruction is executed. The value cannot exceed 32,767. Registers 1-12 can be specified for any RPL operand (that is, when the address of an RPL to be modified is supplied). Register notation for all other operands is restricted to registers 2-12.

- When an *address* is indicated (for example, ACB=acb address) and the macro instruction is a declarative macro instruction (see Figure 1), you can specify any relocatable expression that is valid for an A-type address constant. If the macro instruction is an RPL-based or ACB-based macro instruction, you can use any expression that is valid for an RX-type assembler instruction (such as an LA instruction). Registers 1-12 can be specified for any RPL operand (that is, when an RPL address is being supplied). Register notation for all other operands is restricted to registers 2-12.

If any of the terms used in the format descriptions are unclear, refer to the *OS/VS and DOS/VS Assembler Language* publication.

The valid notation for the operands of the manipulative macro instructions (GENCB, MODCB, SHOWCB, and TESTCB) are not as straightforward. The rules of syntax for the manipulative macro instructions are defined and tabulated in Appendix E.

An example showing how the operand is coded or used may also be included in the operand description. Since there is an example elsewhere showing how the macro instruction as a whole might be coded, an operand example is provided only if the operand is unusually complex, or if its function can be better explained with an example.

Examples

Following the description of the macro instruction are one or more examples. These examples show possible ways that the macro and its operands might be coded.

The way a macro can be specified can often be understood more readily from an example than it can from the assembler format table, since the latter must show all possible ways to specify the macro. A macro that appears to be complex in the assembler format table usually appears far simpler when it is actually coded.

Return of Status Information

All of the macro instructions post return codes in registers and most indicate status information in various control block fields when they are executed. Descriptions of this status information, when applicable, can be found at the end of the macro instruction description. Here you will often find references to Appendixes C and D, where the status information is tabulated.

ACB–Create an Access Method Control Block

The ACB identifies the application program to VTAM and to the teleprocessing network.

Every application program must be defined by the installation before the program can use VTAM to communicate with the terminals throughout the network. The installation does this by creating an APPL entry for the application program in the resource definition table during VTAM definition. The application program's responsibility, then, is to create an ACB that indicates a particular APPL entry. The application program is identified by VTAM when that ACB is opened with the OPEN macro instruction.

When the ACB is opened, requests for connection and then requests for I/O operations can be made (all connection and I/O requests indicate an ACB). When the ACB is closed (with the CLOSE macro instruction), requests can no longer be made, and any connections that were established are broken.

Using the ACB, the application program can provide an address of a list of exit routine addresses. The various routines represented in this list are invoked by VTAM when special events occur, such as error conditions, logon requests, and attention interruptions. The exit list pointed to in the ACB is created with the EXLST (or GENCB) macro instruction.

Using the ACB, the application program can also prevent or allow VTAM to queue logon requests that are directed to the ACB.

Every application program using VTAM must have an ACB. An application program could contain more than one ACB (thus breaking itself down into "subapplications"), but each ACB must indicate a unique APPL entry.

An ACB macro instruction causes an ACB to be built during program assembly. The control block is built on a fullword boundary. (The ACB can also be built during program execution with the GENCB macro instruction. See the GENCB macro for a description of this facility.) The ACB can be modified during program execution with the MODCB macro instruction, but only before it has been opened. The ACB cannot be modified while the ACB is open.

Name	*Operation*	*Operands*
[symbol]	**ACB**	**AM=VTAM** **[,APPLID=address of application program's symbolic name]** **[,PASSWD=password address]** **[,EXLST=exit list address]** **[,MACRF={LOGON | NLOGON}]**

AM=VTAM

Identifies the ACB built by this macro instruction as a VTAM ACB. This operand is required.

APPLID=address of application program's symbolic name

Links the ACB during OPEN processing with a particular APPL entry in the resource definition table. This both identifies the application program to VTAM and associates the application program with any options that might be indicated in the APPL entry.

If you omit this operand, the APPLID field is set to 0. If this field is still set to 0 when OPEN is executed, the job step name specified on the program's EXEC statement (in OS/VS) or the job name (in DOS/VS) is used as the application program's symbolic name.

Format: Expressions involving registers cannot be used with the ACB macro instruction.

Note: *The area pointed to by this operand must begin with a 1-byte length indicator, followed by the application program's symbolic name in EBCDIC. This is the symbolic name that appears on an APPL statement and must conform to the rules for coding this operand described in the appropriate system programmer's guide. The length indicator specifies the length of the name. Any name that is longer than 8 bytes is truncated to 8. You can either pad the name to the right with enough blanks to form an 8-byte name (length indicator of 8), or you can set the length indicator to the actual length of the name you are providing and let VTAM do the padding. In the example at the end of this macro instruction description, the first method is used.*

PASSWD=password address

Allows an application program to associate its ACB with an APPL entry that is password protected. If a password is included in an APPL entry, any application program wanting to link its ACB to that entry must specify the entry's password in the ACB. The two passwords are compared when the application program opens the ACB. If the passwords do not match, the ACB is not opened. (The purpose of this password protection is to prevent a program from running as one of the installation's predefined application programs without the authorization of the installation.) If you omit this operand, the PASSWD field is set to 0.

Format: Expressions involving registers cannot be used with the ACB macro instruction.

Note: *The area pointed to by this operand must begin with a 1-byte length indicator, followed by the EBCDIC password (in alphanumeric characters only). This is the password that is specified using the PRTCT operand of the APPL statement. It must conform to the rules for coding this operand described in the appropriate system programmer's guide. The maximum length is 8 bytes. The truncation and use of the length indicator are the same as described above for the APPLID operand.*

EXLST=exit list address

Links the ACB to an exit list containing addresses of routines to be entered when certain events occur. This list is created by an EXLST (or GENCB) macro instruction. See that macro for descriptions of these events.

More than one ACB can indicate the same exit list. The use of an exit list is optional. If no exit list is used, the application program is not notified that the events described in the EXLST macro instruction occurred.

Format: Expressions involving registers cannot be used with the ACB macro instruction.

MACRF=LOGON|NLOGON

Indicates whether or not the application program wants to receive logons. MACRF=LOGON allows VTAM to queue logons for the application program as they occur. When SETLOGON (OPTCD=START) is issued, the scheduling of the LOGON exit routine begins. SETLOGON (OPTCD=START) will not work unless MACRF=LOGON is specified for the ACB. MACRF=NLOGON indicates that the application program cannot receive logons. Until an application program opens an ACB that specified MACRF=LOGON and issues SETLOGON with either the START or STOP, any logons that may have been directed at the application program are canceled. MACRF=NLOGON serves to notify all application programs issuing INQUIRE (OPTCD=APPSTAT) that logons cannot be directed at the ACB.

A logon is a request issued by (or on behalf of) a terminal and directed at an application program; it in effect asks that application program to connect the application program to the terminal. A queued logon cannot be satisfied until the application program issues an OPNDST macro instruction having an ACCEPT option code in effect for its RPL. This causes the application program to become connected to the terminal.

If the ACB's EXLST operand indicates an exit list containing the address of a LOGON exit routine (see EXLST macro), that routine is entered whenever a logon is queued, providing that SETLOGON (OPTCD=START) is in effect. This routine can issue the OPNDST macro instruction to request connection with the terminal and satisfy the logon.

ACB Fields Not Set by the Application Program

The following ACB fields are set by VTAM when OPEN processing is completed, and cannot be set by the application program. The use of these fields is more fully explained in the OPEN and CLOSE macro instructions.

Field Name	Contents
OFLAGS	Indicates whether or not the ACB has been opened successfully. By specifying OFLAGS=OPEN on a TESTCB macro instruction, you can determine if the ACB has been opened.
ERROR	Indicates why the ACB has not been opened or closed successfully. You can use either SHOWCB or TESTCB to examine the codes in this field. The possible codes, along with their meanings, appear in the OPEN and CLOSE macro instruction descriptions.

Example

```
ACB1      ACB       AM=VTAM,APPLID=NAME,PASSWD=PASFLD,
                    MACRF=LOGON,EXLST=EXLST1
          .
          .
          .
NAME      DC        X'08'
          DC        CL8'PAYROLL'
PASFLD    DC        X'08'
          DC        CL8'SECRET'
```

ACB1 generates an ACB that will be associated with the PAYROLL APPL entry when the ACB is opened. SECRET is the password protecting that APPL entry. MACRF=LOGON means that terminals can issue logon requests to PAYROLL.

When such requests are made, VTAM will note that ACB1 is the ACB providing access to the application program representing PAYROLL, and will invoke the LOGON exit routine indicated in EXLST1.

CHANGE—Change a Terminal's PROC Options or USERFLD Data (Basic Mode Only)

This macro instruction causes modifications to the PROC and USERFLD fields to become effective for a local non-SNA 3270, BSC, or start-stop terminal. Since this means changing the ground rules under which all I/O requests for the terminal are processed, all pending I/O requests for the terminal are canceled when CHANGE is executed.

When an OPNDST macro instruction is executed, the contents of these NIB fields are moved into internal VTAM control blocks. If the application program later wants to change the fields in effect for the terminal, altering the NIB to reflect these changes will not suffice since VTAM is referring to its internal control blocks, not to the NIB. Internal equivalents of the PROC and USERFLD fields must be changed as well. This latter function is provided by the CHANGE macro instruction.

The RPL pointed to in the CHANGE macro instruction must indicate (in its NIB field) the NIB whose PROC or USERFLD field is to be changed. The MODE field of the NIB to be changed must specify BASIC. The CID of the terminal must be set in the NIB's CID field. RPL fields (but not the NIB fields) can be set with the CHANGE macro instruction itself.

To change the NIB fields, this procedure should be followed:

1. Modify the fields in the NIB with MODCB. For example:

```
MODCB   AM=VTAM,NIB=NIB4,USERFLD=NYC,
        PROC=(TRANS,CONFTXT,MONITOR)
```

2. Issue the CHANGE macro instruction to make these changes effective. CHANGE can simultaneously be used to make the RPL's NIB field point to the modified NIB, if it does not already do so:

```
CHANGE  RPL=RPL1,NIB=NIB4
```

Name	*Operation*	*Operands*
[symbol]	CHANGE	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address
Indicates the RPL whose NIB field contains the address of the NIB that has been modified.

rpl field name=new value
Indicates an RPL field to be modified, and the new value that is to be contained within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the CHANGE macro instruction.

Format: For *rpl field name,* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register.

The following RPL operands apply to a CHANGE macro instruction:

ACB=acb address
Indicates the ACB used when the terminal was connected.

NIB=nib address
Indicates the NIB whose CID field identifies the terminal whose PROC or USERFLD attributes are being changed.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) CHANGE macro instruction is completed. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When the SYN option code is set, control is returned to the application program when the CHANGE operation has been completed. When ASY is set, control is returned as soon as VTAM has accepted the request. Once the operation is completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB-EXIT field.

OPTCD=CS | CA
When CA is set, data obtained from the terminal can satisfy a READ (OPTCD=ANY or OPTCD=SPEC) macro instruction. When CS is set, only READ (OPTCD=SPEC) macro instructions can obtain data from the terminal.

Return of Status Information

After the CHANGE operation is completed, the following RPL fields are set:

The value 25 (decimal) is set in the REQ field, indicating a CHANGE request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

CHECK–Check Request Status

When asynchronous handling has been specified for a request (ASY option code in effect), the application program receives control when the request has been accepted by VTAM and the requested operation has been scheduled. A CHECK macro instruction must be issued for the RPL used for the request. (CHECK should not be issued for synchronous requests.)

When CHECK is executed for an RPL that specifies an internal or external ECB, the following actions occur:

- The RPL being checked is marked available for reuse by another request. (CHECK is the only way this can be done for asynchronous requests.)
- If the operation being checked has not been completed, execution of the application program is suspended until it is completed.
- If the operation being checked has been completed successfully, control is returned to the application program.
- If the operation being checked has been completed unsuccessfully, the LERAD or SYNAD exit routine is invoked, if available.
- The ECB (internal or external) is cleared before control is returned to the application program. (Clearing the ECB is necessary before the next or first RPL-based macro instruction using this RPL is issued.)

  Note: *The ECB specified in the RPL-based macro instruction must not be cleared between the time the RPL-based macro instruction is issued and the corresponding check is issued. If the ECB is cleared during this interval, control may not be returned to the application program after the CHECK is issued.*

When CHECK is executed in an RPL exit routine, the following actions occur:

- The RPL being checked is marked available for reuse by another request.
- If the operation being checked was completed unsuccessfully, the LERAD or SYNAD exit routine is invoked, if available.

Note: *When a RPL exit routine is used, the CHECK macro instruction should be issued only in the RPL exit routine. If the CHECK is issued outside of the exit routine and the CHECK is executed before the RPL exit routine is invoked, the CHECK will fail with a return code of X'18' in register 0.*

See the description of the CHECK macro instruction in the *Macro Language Guide* for more information.

Name	*Operation*	*Operands*
[symbol]	**CHECK**	**RPL=rpl address**

RPL=rpl address
Indicates the address of the RPL associated with the connection or I/O request whose completion status is being checked.

Format: Register notation (for registers 1-12) is valid.

Note: *See the ECB and EXIT operands in the RPL macro instruction description for more information about the RPL exit routine and the ECB.*

Example

```
CHK1          CHECK        RPL=RPL1
```

If CHK1 is in the routine indicated by RPL1's EXIT field, and the operation requested via RPL1 ends with a logical or other error, the LERAD or SYNAD exit routine is scheduled.

If there is no RPL exit routine for RPL1, CHK1 causes program execution to stop until the operation requested via RPL1 has ended. If the operation ends with a logical or other error, CHK1 causes the LERAD or SYNAD exit routine to be invoked.

Return of Status Information

When CHECK processing has been completed, registers 0 and 15 are set as indicated in Appendix C. If an error occurred and a LERAD or SYNAD exit routine was invoked, these registers contain the values set in them by the exit routine. Otherwise, VTAM places a general return code in register 15 and a recovery action return code in register 0 (see Figures C-4 and C-5 in Appendix C).

CLOSE–Close One or More ACBs

There are three significant results of executing the CLOSE macro instruction:

- VTAM no longer accepts any connection or I/O requests that refer to the ACB specified in the CLOSE macro. This ACB is effectively disconnected from VTAM.
- VTAM no longer maintains the association between the APPL entry in the resource definition table and the ACB specified in this macro instruction. CLSDST (PASS) logons that are directed towards the application program cannot cause the LOGON exit routine to be scheduled, but are queued awaiting the next OPEN. Insofar as terminals requesting logon are concerned, the portion of the application program represented by the ACB ceases to exist when CLOSE is executed.
- VTAM breaks every connection that exists between the ACB and other terminals. Before CLOSE breaks a connection, all I/O activity is stopped and all pending I/O requests are canceled. (For logical units, a Clear command is issued, and for BSC and start-stop terminals, a RESET operation is performed.)

The CLOSE macro instruction can be applied to more than one ACB. CLOSE must be issued in the main program or in the LERAD or SYNAD exit routine if the routine has been entered directly from the main program. Never issue CLOSE in the RPL exit routine or in any of the other EXLST exit routines.

In OS/VS, where the privileged user can manage multiple tasks in the same application program, all I/O requests must be completed before CLOSE can be issued in the main part of the mother task.

Name	*Operation*	*Operands*
[symbol]	**CLOSE**	**acb address[, acb address] ...**
This form of CLOSE is valid in DOS/VS only.		
[symbol]	**CLOSE**	**(acb address[,, acb address] ...)**
This form of CLOSE is valid in OS/VS only.		

acb address

Indicates the ACB that is to be disconnected from VTAM.

Format: If more than one ACB is specified, separate each with a comma if the program is going to be run under DOS/VS. Separate each ACB address with *two* commas if the program is going to be run under OS/VS. The parentheses for the OS/VS CLOSE can be omitted if only one address is coded.

Note: *One CLOSE macro instruction can be issued to close VSAM ACBs in addition to VTAM ACBs. DOS/VS users can also include DTFs with this macro instruction, and OS/VS users can also include DCBs.*

Example

CLOSE123	CLOSE	ACB1,ACB2,(7)	(DOS/VS)
CLOSE123	CLOSE	(ACB1,,ACB2,,(7))	(OS/VS)

CLOSE123 closes ACB1, ACB2, and the ACB whose address is in register 7. All terminals connected via these ACBs are disconnected.

Return of Status Information

When control is returned to the instruction following the CLOSE macro, register 15 indicates whether or not the CLOSE processing has been completed successfully. Successful completion (meaning that all ACBs specified in the maco instruction have been disconnected from VTAM) is indicated by a return coe of 0 (for DOS/VS users, register 15 is left unmodified). Unsuccessful completion is indicated by the following register 15 values:

For DOS/VS	Meaning
nonzero	One or more ACBs (or DTFs or VSAM ACBs) were not successfully closed.

For OS/VS	Meaning
04	One or more ACBs were not successfully closed. Depending on the type of error, the OFLAGS field may indicate that the ACB is closed even though CLOSE has failed (for example, the ACB may never have been opened).
08	One or more ACBs were not successfully closed. Inspect the ERROR field for the cause of the failure. Another CLOSE macro instruction may be issued.
0C	One or more ACBs were not successfully closed. Another CLOSE macro instruction may not be issued.

If unsuccessful completion is indicated, the application program can examine the OFLAGS field in each ACB to determine which ACB was not closed. If you use the OFLAGS=OPEN operand on a TESTCB macro instruction, an "equal" PSW condition code will result if the ACB was not closed.

For each ACB, you can use either the SHOWCB or TESTCB macro instruction to check the ERROR field and determine the cause of the error. For example:

```
SHOWCB      AM=VTAM,ACB=ACB1,FIELDS=ERROR,AREA=SHOWIT,
            LENGTH=4
```

Note: *If the ACB address specified in the CLOSE macro instruction does not indicate an ACB or lies beyond the addressable range of your application program,* nothing *is posted in the ACB's ERROR field.*

The value set in the ERROR field indicates the specific nature of the error encountered by CLOSE (all values except 48 apply to both DOS/VS and OS/VS):

0	CLOSE successfully closed the ACB.
4	A CLOSE macro instruction has already been successfully issued for this ACB (or the ACB has never been opened in the first place).
42 (66)	The ACB has been closed, but an apparent system error has prevented the successful disconnection of one or more of the terminals connected to the application program. The fault is VTAM's, and IBM program systems representatives should be

consulted. The terminals that could not be disconnected are not available to other application programs, and terminals for which you were requesting connection when CLOSE was executed will likewise be unavailable when they are released to you. You can notify the network operator (during program execution) of the situation so that the operator can make the terminals available to other application programs.

46 (70) CLOSE was not issued in the main program. OPEN and CLOSE cannot be issued in any exit routine except LERAD or SYNAD.

48 (72) CLOSE was not issued in a job step task or in a subtask within the VTAM region (OS/VS1 and OS/VS2 SVS only).

4C (76) This application program is authorized to issue network operator commands and receive VTAM messages. A CLOSE was issued, but there are unreceived messages still queued for it or VTAM is waiting for a reply or both. VTAM considers the application program to be in a quiesced state, and only RCVCMD macro instructions to receive the queued messages or SENDCMD macro instructions containing a REPLY command to send a reply to VTAM can be issued. Any RCVCMD macro instructions issued while the application program is in this quiesced state will not be queued by VTAM. Therefore, it is recommended that they be issued with NQ specified for the OPTCD operand and processed immediately. Once a return code indicates that there are no more messages waiting, a second CLOSE should complete normally.

50 (80) VTAM is no longer included as part of the operating system.

70 (112) CLOSE was issued while the program was in the process of terminating abnormally. The CLOSE is not necessary since the ACB will be closed by VTAM when the task terminates.

BC (188) The ACB is in the process of being opened or is in the process of being closed by another CLOSE request.

CLSDST—Disconnect Terminals from the Application Program

The CLSDST (close destination) macro instruction requests VTAM to break or not make a connection between the application program and a specified terminal (logical unit, a BSC or start-stop terminal, or a local non-SNA 3270). CLSDST cancels any pending I/O requests for the terminal, and any unread data from the terminal is lost.

The terminal to be disconnected is specified with either the ARG field or the NIB field of CLSDST's RPL:

- If the ARG field contains the CID of a terminal, that terminal is disconnected.
- If the NIB field contains the address of a NIB, the terminal whose symbolic name has been placed in that NIB's NAME field is disconnected.

(The RPL cannot contain both a CID and a pointer to a NIB, because the ARG and NIB fields occupy the same area in the RPL control block.)

Using a CID is easier following normal communications with te terminal, since the CID is used by all of the I/O requests and thus should be readily available. Using a NIB address and symbolic name is necessary if you are issuing CLSDST for a terminal that was never connected to your application program. For example, you must issue CLSDST in order to reject a logon request, and you can cancel a pending OPNDST (OPTCD=ACCEPT) macro instruction by issuing CLSDST. In both of these situations, only the terminal's symbolic name is available to you.

For BSC and start-stop dial terminals, CLSDST with OPTCD=RELEASE causes a dial-line disconnection only if no other application program has requested connection with the terminal. For dial logical units, CLSDST with OPTCD=RELEASE will cause a dial-line disconnection only if the logical unit definition supplied by the user in the PU statement specified that the disconnection should take place. See the appropriate system programmer's guide for more information.

If, at the time CLSDST is executed, VTAM buffers hold data from the terminal, the data is not saved for the next application program that becomes connected to the terminal, but is discarded.

The CLSDST macro instruction can optionally be used to request that VTAM reconnect the terminal to another application program (specified by you) in addition to disconnecting it. This option is implemented by setting the PASS option code in CLSDST's RPL. If this option is used (it must be authorized by the installation), VTAM first disconnects the terminal and then generates a logon for it. Your application program must indicate which application program is to receive the logon. A logon mode name may be specified in the NIB (if none is specified, the default logon mode name for the terminal is used). A logon message from a data area in your program can also be sent with the logon. (The data area containing the logon message can be reused as soon as CLSDST has been completed.)

If a logon is going to be generated after the disconnection, the RPL's PASS option code must be set, and the RPL's AAREA field must point to the symbolic name of the receiving application program. This name must be placed in an 8-byte field, left justified, and padded to the right with blanks. If a logon message is also to be sent with the logon, the AREA and RECLEN fields must indicate the location and length of the message. If a message is not to be sent, the RECLEN field must be set to 0.

CLSDST (OPTCD=PASS) will fail if the receiving application program has not been activated, has not yet opened its ACB, has opened its ACB with MACRF=NLOGON specified, or has issued SETLOGON (OPTCD=QUIESCE) to close its logon queue. However, CLSDST (OPTCD=PASS) will cause a logon to be queued if the receiving application program has issued SETLOGON (OPTCD=STOP), even though this indicates that the application program temporarily does not want any logons directed at it. For logons generated by CLSDST (OPTCD=PASS), an INQUIRE (OPTCD=APPSTAT) macro instruction normally should be issued before CLSDST (OPTCD=PASS) is issued. The return code from INQUIRE indicates the status of the receiving program.

If the RELEASE option code is used instead of the PASS option code, the terminal is simply disconnected as far as as the application program is concerned. If another application program has requested connection to the terminal, or if the user indicated. during VTAM definition that automatic logons are to be generated, VTAM reconnects the terminal to the appropriate application program.

If an application program has completed its processing and is ready to disconnect *all* of the terminals connected to it, a CLSDST macro instruction for each terminal need not be used. The CLOSE macro instruction may be used; it disconnects all of the terminals connected via a given ACB (as though CLSDST with the RELEASE option had been issued for each one). See the description of the CLSDST macro instructions in the *Macro Language Guide* for performance considerations.

Name	*Operation*	*Operands*
[symbol]	CLSDST	RPL=rpl address [,rpl field name=new value] . . .

RPL=rpl address
Indicates the location of the RPL to be used during CLSDST processing. Either the ARG field of this RPL must contain a terminal's CID or the NIB field must be set to point to the NIB containing the symbolic name of the terminal.

rpl field name=new value
Indicates an RPL field to be modified and the new value that is to be contained within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the CLSDST macro instruction.

Format: For *rpl field name*, code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. ARG can also be coded. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register. The value supplied with the ARG keyword must indicate a register.

The following RPL operands apply to a CLSDST macro instruction:

ACB=acb address
Indicates the ACB from which the terminal is to be disconnected.

NIB=nib address
If OPTCD=RELEASE is specified, it indicates the NIB whose NAME field identifies the terminal to be disconnected. If OPTCD=PASS is specified, it indicates a NIB whose NAME field identifies the terminal that is to be disconnected and reconnected to another application program and, optionally, whose LOGMODE field specifies the session parameters to be used for the reconnection. If the NIB field does not indicate a NIB address, the ARG field must contain the terminal's CID.

ARG=(register)
Indicates the register that contains the CID of the terminal to be disconnected. This register notation must be used if the CID is to be placed into the ARG field with this CLSDST macro instruction. ARG and NIB provide two mutually exclusive methods of identifying the terminal.

AREA=address of logon message
Indicates the location of the data to be sent to the application program receiving the terminal. The content and format of the data is determined by the sending and receiving application programs. The logon message is equivalent to the data portion of an Initiate Self command or a character-coded logon. A logon message is sent only if OPTCD=PASS is set.

RECLEN=length of logon message
Indicates how many bytes of data are to be sent to the application program receiving the terminal. No data is sent if RECLEN is set to 0.

AAREA=address of receiver's symbolic name
Indicates the name of the application program that is to be connected to the terminal you are disconnecting. You can specify the application program that is to receive the terminal only if OPTCD=PASS is set. The name must be 8 bytes long and padded to the right with blanks.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) CLSDST macro instruction is completed. The macro instruction is completed when I/O has been canceled and the terminal has been disconnected; completion does not depend on the receiving application program's issuing OPNDST. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When SYN is set, control is returned to the application program when the CLSDST operation is completed. When ASY is set, control is returned as soon as VTAM has accepted the CLSDST request. Once the operation has been completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB-EXIT field.

OPTCD=RELEASE | PASS
When RELEASE is set, VTAM determines the identity of the terminal's next owner (if any). When PASS is set, a logon is directed at the application program whose symbolic name is indicated in the AAREA field of the RPL used by CLSDST. If the AREA and RECLEN fields are also set, a logon message is sent to the application program. The use of PASS must be authorized by the installation.

Examples

```
CL1        CLSDST   RPL=RPL1,
                    ACB=ACB1,
                    NIB=NIB3,                      (LOGICAL UNIT TO BE DISCONNECTED)
                    AAREA=APPLNAME,                (APPLICATION TO RECEIVE LOGON)
                    AREA=LGNMSG,RECLEN=60,         (LOGON MODE AND MESSAGE)
                    ECB=POSTIT1,OPTCD=(ASY,PASS)
             .
             .
             .
POSTIT1    DS       F
NIB3       NIB      NAME=LU1,LOGMODE=BATCH
APPLNAME   DC       CL8'PLOTTER'
LGNMSG     DC       CL60'LOGON FROM LU1'
```

CL1 disconnects the logical unit represented in NIB3 (LU1) and generates a logon for it; the logon is directed at the application program named PLOTTER. This macro instruction also sends a logon mode (BATCH) and 60 bytes of information containing a logon message (LGNMSG) with the logon. The logon message and the session parameters that VTAM derives from the logon mode can be accessed (via INQUIRE) prior to OPNDST by the application program receiving the logon. If it is desired to send the actual logon mode name (BATCH, in this example) to the application program receiving the logon, the logon name could be specified as data as part of the original LGNMSG; the NIB LOGMODE parameter would still be required.

```
CL2        CLSDST   RPL=RPL2,
                    ARG=(3),                       (TERMINAL TO BE DISCONNECTED)
                    ECB=POSTIT2,OPTCD=(ASY,RELEASE)
```

CL2 disconnects the terminal whose CID has been placed in register 3. Unlike the first example above, CL2 does not generate a logon request for a specified application program, nor does it send a logon mode or a logon message.

```
CL3        CLSDST   RPL=RPL3,
                    NIB=NIB6,                      (LOGICAL UNIT TO BE DISCONNECTED)
                    AAREA=APPLNAME,                (APPLICATION TO RECEIVE LOGON REQUESTS)
                    RECLEN=0,                      (NO LOGON MODE OR MESSAGE)
                    ECB=POSTIT3,OPTCD=(ASY,PASS)
             .
             .
             .
APPLNAME   DC       CL8'PLOTTER'
POSTIT3    DC       F'0'
NIB6       NIB      NAME=LU3
```

CL3 disconnects the logical unit represented by NIB6 (LU3), and generates a logon for it that is directed at the PLOTTER application program. Since the RECLEN field is being set to 0, no logon message is sent to PLOTTER. The default logon mode of 8 blanks is assumed.

Return of Status Information

After the CLSDST operation is completed, the following RPL fields are set:

The value 31 (decimal) is set in the REQ field, indicating a CLSDST request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

DO–Initiate LDO-Specified I/O Operations (Basic Mode Only)

If an application program uses logical device orders (LDOs) to request I/O operations, it must use the DO macro instruction to initiate the operations. The special I/O operations initiated with DO are described in the LDO macro instruction.

The user of the DO macro instruction specifies an RPL whose AREA field contains the address of an LDO or list of LDOs, and whose ARG field contains the CID of the BSC or start-stop terminal that is to be the object of the I/O operations. Changes to the RPL can be specified in the DO macro instruction itself.

When DO is completed, the AAREA field of the RPL indicates the address of the last LDO used by DO. If an error occurs, AAREA contains the address of the LDO that was being processed when the error occurred.

Name	*Operation*	*Operands*
[symbol]	**DO**	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address
Indicates the location of the RPL whose AREA field contains the address of an LDO or list of LDOs to be used, and whose ARG field contains the CID of the terminal that is to be the object of these LDOs.

rpl field name=new value
Indicates a field of the RPL to be modified and the new value that is to be contained within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the DO macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds with the RPL field to be modified. ARG can also be coded. The *new value* can be any value that could have been supplied with the keyword had the operand been issued in an RPL macro instruction, or it can indicate a register. The value supplied for the ARG keyword must indicate a register.

The following RPL operands apply to the DO macro instruction:

ACB=acb address
Indicates the ACB that was used when the terminal was connected.

ARG=(register)
Indicates the register containing the CID of the terminal. This register notation must be used when the CID is placed into the ARG field with this DO macro instruction.

AREA=ldo address
Indicates the LDO or list of LDOs to be used by this macro instruction.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken when an asynchronous (OPTCD=ASY) DO macro instruction is completed. The macro instruction is completed when the last LDO has been processed. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When the SYN option code is set, control is returned to the application program when the DO operation has been completed. When ASY is set, control is returned as soon as VTAM has accepted the request. Once the DO operation has been completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB=EXIT field.

OPTCD=CS | CA
When CA is set, data obtained from the terminal can satisfy a READ (OPTCD=ANY or OPTCD=SPEC) macro instruction. When CS is set, only READ (OPTCD=SPEC) macro instructions can obtain data from the terminal. See the RPL macro instruction for more information.

Example

```
DOLDO      DO         RPL=RPL1,
                      AREA=(2),ARG=(3),
                      EXIT=DONE,OPTCD=(SPEC,ASY)
```

DOLDO initiates whatever operations are indicated by the LDO (or list of LDOs) pointed to by register 2. In this example, register 3 must contain the CID of the terminal to be involved in the LDO-specified I/O operation or operations. Since the ASY option code is specified, control is returned to the instruction following DOLDO before the operation is actually performed. Since EXIT is specified, the routine located at DONE will be scheduled when the DO macro instruction is completed.

Return of Status Information

Once DO processing is finished, the following RPL fields are set:

The address of the last LDO used by DO is placed in the AAREA field.

When a NIB is used by OPNDST, the user has the option of specifying an arbitrary value in the USERFLD field of that NIB. When the DO macro instruction is subsequently issued for the terminal associated with that NIB, whatever was placed in USERFLD by the user is placed in the USER field of the RPL by VTAM.

If DO is processing a READ LDO or READBUF LDO, the RECLEN field is set to indicate the number of bytes of data obtained from the terminal.

The value 19 (decimal) is set in the REQ field, indicating a DO request.

If DO is processing a READ or WRITE LDO, the SENSE field is set as indicated in Appendix C.

If DO is processing a READ LDO, the FDBK field is set as indicated in Appendix C.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

EXECRPL–Execute a Request

When a request fails for a temporary reason and the request might succeed if reissued, VTAM returns a recovery action return code of 8 in register 0 and in the RPL's RTNCD field. The portion of the application program receiving control (the SYNAD exit routine or the next sequential instrucion) has the address of the RPL available to it in register 1. The program can issue an EXECRPL macro instruction to retry the request without having to modify the request's RPL.

The operation performed by EXECRPL depends on the request code that is set in the RPL's REQ field. If the REQ field indicates a RECEIVE request, for example, the effect of EXECRPL is identical to that of a RECEIVE macro instruction. (The REQ field is described in the RPL macro instruction.) EXECRPL can be used to execute any RPL-based request except CHECK or another EXECRPL macro.

When EXECRPL is used for its intended purpose–that is, to reexecute a request *that has failed with a recovery action return code of 8*–the application program need not concern itself with the RPL contents when EXECRPL is issued. But if EXECRPL is used to retry a request that has failed with some other recovery action return code (or to repeat a request that has not failed at all), close attention must be paid to RPL fields that can be set by the application program and then modified by VTAM while the request is being processed.

For example: The RPL's RESPOND field for a SEND request is set by the application program (to indicate whether a response is to be returned) and is then reset by VTAM (to indicate the type of response returned). EXECRPL should not be issued for this RPL unless the RESPOND field is reset to its intended setting.

Figure 10 at the end of the RPL macro instruction description identifies those fields that can be set by the application program and then reset by VTAM. These fields are identified with an 'AV' under the appropriate macro instruction.

Name	*Operation*	*Operands*
[symbol]	**EXECRPL**	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address
Indicates the location of the RPL defining the request to be reexecuted.

rpl field name=new value
Indicates a field of the RPL to be modified and the new value that is to be contained within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the EXECRPL macro instruction.

Format: For *rpl field name*, code the keyword of the RPL macro instruction operand that corresponds with the RPL field to be modified. The *new value* can be any value that could have been supplied with the keyword had the operand been issued in an RPL macro instruction, or it can indicate a register.

Example

RETRY1 EXECRPL RPL=(1)

A SYNAD exit routine has been entered for a retriable error (register 0 is set to 8). The request is reexecuted as defined by the current contents of the RPL.

Return of Status Information

Once the EXECRPL macro instruction is completed, the action taken by VTAM depends on the type of request that EXECRPL has processed. The manner in which the application program is notified of completion (ECB or EXIT), the RPL fields and return codes that are returned, and the data areas (if any) that are used depend on the contents of the RPL when EXECRPL was executed. If the request is successfully accepted or completed, then registers 0 and 15 at the next sequential instruction after EXECRPL are set exactly as expected when the original request was issued.

EXLST–Create an Exit List

The EXLST macro instruction builds a list of routine addresses during program assembly. Each operand in this macro instruction represents an event for which an exit routine is invoked by VTAM. The address supplied for each operand indicates the user-written routine to be given control when the event that it handles occurs. The SYNAD operand supplies the address of a routine that handles exception conditions (other than logical errors), the ATTN operand supplies the address of an attention-interruption handler, and so forth.

When you examine your program listing, you may discover that the assembler has reserved space for exit list addresses that you never specified. Unspecified exits will not, however, be used by VTAM, and you cannot use MODCB to insert an address in a field you never specified in the EXLST (or GENCB) macro instruction. An address of 0 can never be specified.

Two of the exit routines are invoked by events initiated *within* the application program. The LERAD exit routine is invoked when a request results in a logical error; the SYNAD exit routine is invoked when a request results in other errors. If the error involves a synchronous request (one for which the SYN option code is in effect), the exit routine is invoked when the error condition is detected. If the error involves an asynchronous request (ASY option code) that has been accepted, the exit routine is not invoked until a CHECK macro instruction is issued for the request. (For requests of either type that are not accepted, the exit routine is invoked when the error is detected.)

The other exit routines are invoked as a result of an event initiated *outside* of the application program. These exit routines are scheduled at the time the event occurs. One routine, LOGON, falls into both categories. The programmer should be aware that if any synchronous requests are made in these exit routines, neither the exit routine nor the main part of the application program can receive control while the request is being completed.

When the LERAD and SYNAD exit routines are invoked, register 1 contains the address of the failing request's RPL. When the other exit routines are invoked, register 1 contains the address of a parameter list. The contents of the parameter lists vary somewhat between exit routines. The parameter lists are described in detail below and are summarized in Figure 3.

For all exit routines except LERAD and SYNAD, the last instruction must be a branch to the VTAM address that is in register 14 when the routine receives control. (For LERAD and SYNAD, this branch is required only if LERAD or SYNAD is invoked by a macro instruction issued within an RPL exit routine or other EXLST exit routine.) The exit routines are not provided with a save area for the general purpose registers. The application program may use and change registers as desired, but the register 14 address must be saved. The address of the exit list created by the EXLST macro instruction is placed in the EXLST field of an ACB by the application program (see the ACB macro instruction for details). More than one ACB can point to the same exit list. In this situation, however, the routines indicated in the exit list should be reenterable.

The address of an EXLST containing a DFASY, RESP, or SCIP exit routine address can also be placed in the EXLST field of a NIB. These NIB-oriented exit routines are scheduled when input arrives from the logical unit represented by the NIB. If, for example, DFASY input arrives from a logical unit, VTAM first determines

Exit-Routine	Register 1 Parameter List				
	1st Word	2nd Word	3rd Word	4th Word	5th Word
LERAD	None (Register 1 contains the RPL address for the request that failed)				
SYNAD	None (Register 1 contains the RPL address for the request that failed)				
DFASY	ACB address	CID	USERFLD data	Unused	Read-only RPL address
RESP	ACB address	CID	USERFLD data	Unused	Read-only RPL address
SCIP	ACB address	CID	USERFLD data	Unused	Read-only RPL address
TPEND	ACB address	Reason-terminated code			
RELREQ	ACB address	Address of the terminal's symbolic name			
LOGON	ACB address	Address of the terminal's symbolic name	Unused	Length of logon message	
LOSTERM	ACB address	CID	USERFLD data	Reason-lost code	
ATTN	ACB address	CID	USERFLD data		

Figure 3. Parameter List for the EXLST Exit Routines

whether a DFASY exit routine was indicated when that logical unit was connected. If no NIB-oriented DFASY exit routine exists, VTAM determines whether an eligible RECEIVE is available or an ACB-oriented DFASY exit routine was indicated when the ACB was opened. If so, the RECEIVE is completed or the exit routine is scheduled.

A few of the exit routines apply only to BSC and start-stop terminals or only to logical units. These are noted below as "basic-mode only" or "record-mode only" respectively.

Note: *Only those exit routines that can be recognized by VTAM may be specified with the VTAM EXLST macro instruction. In addition, VTAM requests are permitted only in VTAM asynchronous exit routines. See the* Macro Language Guide *for more information about using these exit routines.*

Name	*Operation*	*Operands*
[symbol]	**EXLST**	**AM=VTAM** **[,LERAD=** **[,SYNAD=** **[,DFASY=** **[,RESP=** **[,SCIP=** **[,TPEND=** **[,RELREQ=** **[,LOGON=** **[,LOSTERM=** **[,ATTN=** **exit-routine address]**

AM=VTAM

Identifies the exit list generated by this macro instruction as a VTAM exit list (as distinguished from a VSAM exit list). This operand is required.

LERAD=exit routine address

Indicates the address of a routine that will be entered when the application program makes a connection or I/O request that results in a *logical error.*

Generally, logical errors result when an RPL-based request is made that is inherently contradictory–like attempting to read data from an output-only device. (Errors that occur because of hardware malfunctions, for example, are not logical errors; they are handled by the SYNAD exit routine.)

If the SYN option code is in effect when the error occurs or if the request cannot be accepted due to a logical error, the LERAD exit routine is entered immediately; otherwise if the ASY option code is in effect, the routine is not scheduled until a CHECK macro instruction is issued for the operation in which the error occurred. One exception: If the ASY option code is set, the request is accepted by VTAM, and then VTAM determines that it cannot post the RPL (perhaps because the ACB has been overwritten), VTAM abnormally terminates the application program.

Before the LERAD exit routine is given control, VTAM sets a recovery action return code of 20 or 24 (decimal) in register 0 and in the RTNCD field of the RPL and sets a specific error return code in the FDBK2 field indicating the specific cause of the error. These codes are explained in Appendix C.

If the application program has no LERAD exit routine and a logical error occurs, VTAM simply returns control to the next sequential instruction. VTAM places a return code of 4 in register 15 and a recovery action return code of 20 or 24 (decimal) in register 0 and in the RTNCD field of the RPL. It also sets a specific error return code in the FDBK2 field of the RPL indicating the specific cause of the error. These codes are explained in Appendix C.

If the application program issues any RPL-based requests in both the main program and in the exit routines, the LERAD exit routine may be reentered by VTAM. The routine may likewise be reentered if any RPL-based requests are issued in the

LERAD routine itself. In these situations, you must insure that the exit routine is reenterable.

When the LERAD exit routine returns control to VTAM, VTAM leaves registers 0 and 15 intact so that the routine can pass information back in these registers to the main part of the application program.

Registers Upon Entry: When the LERAD routine receives control, the general purpose registers contain the following:

Register 0: A recovery action return code (see Appendix C).

Register 1: The address of the RPL associated with the request. If the recovery action return code in register 0 is set to 24 (decimal), VTAM was unable to place an indicator in the FDBK2 field specifying the reason for the error. This happens in three cases: A macro has been issued whose RPL is already in use, CHECK has been issued for a request whose RPL exit routine has not yet been scheduled, or an invalid RPL was specified (for example, the RPL address is invalid or the RPL is overlaid). See Appendix C for a description of the return codes placed in FDBK2.

Register 13: The address of an 18-word save area supplied by you when the macro instruction that caused the logical error was issued. If the exit routine is going to return control via register 14, it must not change anything in the save area. This means that if any macro instructions are issued in the exit routine, register 13 must first be loaded with the address of a new save area. Furthermore, before control is returned via register 14, register 13 must be restored with the value it had when the exit routine was invoked.

Register 14: The address in VTAM to which the LERAD exit routine can branch when it has finished processing. When the exit routine branches to this address, VTAM handles the returning of control to the next sequential instruction in the application program following the request (or following the CHECK macro instruction issued for the request). The LERAD exit routine can branch to any part of the main program because the routine is executed under the same OS/VS or DOS/VS task control block as the main program. (Care should be taken, however, to eventually return to the register 14 address if LERAD was entered from an RPL-based request issued in another exit routine.) If the routine returns control to the next sequential instruction by branching on the register 14 address, VTAM restores the registers from the save area whose address is in register 13.

Register 15: The address of the LERAD routine.

The contents of the remaining registers (2-12) are unmodified; whatever was in them when the macro instruction was issued is still there.

See the *VTAM Macro Language Guide* for more information on the use of a LERAD exit routine.

SYNAD=exit routine address

Indicates the address of a routine that is entered if an unrecoverable input or output error (physical error) or other unusual condition occurs during an I/O operation. (Errors that result from invalid requests are handled by the LERAD exit routine.) The SYNAD exit routine is entered for all recovery action return codes of 4, 8, 12, and 16 (decimal).

If the SYN option code is in effect when the error occurs or if the request cannot be accepted, the SYNAD exit routine is entered immediately; otherwise, if the ASY option code is in effect, the routine is not invoked until a CHECK macro is issued for the operation in which the error occurred.

The SYNAD exit routine can examine the REQ field of the RPL and determine the type of request that caused the routine to be invoked. Each RPL-based macro instruction (except CHECK and EXECRPL) has its own unique REQ code. These codes are listed in the description of the RPL macro instruction. The SYNAD exit routine can analyze the FDBK2 field and attempt to recover from the error.

If the application program has no SYNAD exit routine and a physical error occurs, VTAM simply returns control to the next sequential instruction with return codes in registers 0 and 15.

If your application program issues RPL-based requests in both the main program and the exit routines, the SYNAD exit routine may be reentered by VTAM. The routine may likewise be reentered if RPL-based requests are issued in the exit routine itself. In these situations, you must ensure that the exit routine is reenterable.

When the SYNAD exit routine returns control to VTAM, VTAM leaves registers 0 and 15 intact; this enables the routine to pass information back in these registers to the main part of the application program.

Registers Upon Entry: When the SYNAD routine receives control, the general purpose registers contain the following:

Register 0: A recovery action return code (see Appendix C).

Register 1: The address of the RPL associated with the request.

Register 13: The address of an 18-word save area supplied by you when the macro instruction that caused the physical error was issued. If the exit routine is going to return control via register 14, it must not change anything in the save area. This means that if any macro instructions are issued in the exit routine, register 13 must first be loaded with the address of a new save area. Furthermore, before control is returned via register 14, register 13 must be restored with the value it had when the exit routine was invoked.

Register 14: The address in VTAM to which the SYNAD exit routine can branch when it has finished processing. When the exit routine branches to this address, VTAM handles the return of control to the next sequential instruction following the request (or following the CHECK macro issued for the request). The SYNAD exit routine can branch to any part of the main program. (Care should be taken, however, to eventually return to the register 14 address if SYNAD was entered from an RPL-based request issued in another exit routine.) If the application program eventually returns to the next sequential instruction by branching on the register 14 address, VTAM restores the registers from the same area whose address is in register 13.

Register 15: The address of the SYNAD routine.

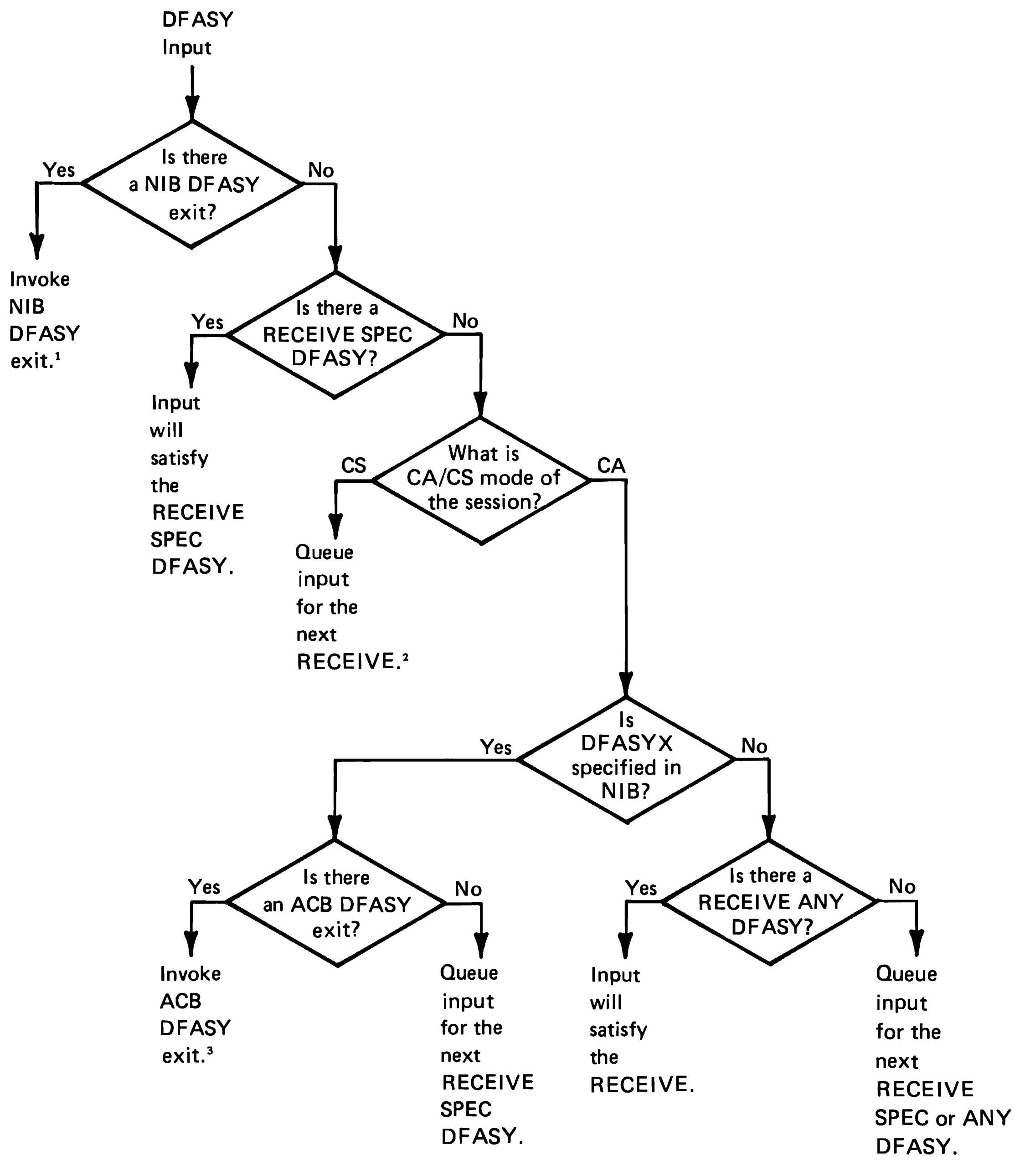
The contents of the remaining registers (2-12) are unmodified; whatever was in them when the macro instruction was issued is still there.

See the *VTAM Macro Language Guide* for more information on the use of SYNAD.

DFASY=exit routine address (Record mode only)

The EXLST containing a DFASY exit routine address can be pointed to by a NIB, as well as by an ACB (see the EXLST operand of the NIB macro instruction).

The DFASY operand indicates the address of a routine to be entered when expedited-flow messages (such as QEC, RELQ, and RSHUTD commands) arrive from a logical unit. VTAM handles the input in this manner:



Notes: [1] The exit routine is scheduled if no other exit routine (including the NIB DFASY exit routine) is currently running; if there is an exit routine running, the input is queued for the NIB DFASY exit routine.

[2] The input will satisfy a RECEIVE SPEC DFASY. The input can also be obtained by a RECEIVE ANY DFASY if the receive mode has been switched to CA.

[3] The exit routine is scheduled if no other exit routine (including the ACB DFASY exit routine) is currently running; if there is an exit routine running, the input is queued for the ACB DFASY exit routine.

The DFASY exit routine provides a way for VTAM to notify the application program that expedited-flow messages have arrived. The application program could maintain an active RECEIVE macro instruction for this purpose, but the RECEIVE requires that an active RPL be committed before the input arrives, while the DFASY exit routine does not.

No RECEIVE is issued in the DFASY exit routine to obtain the message. Instead, the exit routine is passed the address of a read-only RPL. The read-only RPL fields are set as though a RECEIVE macro instruction (RTYPE=DFASY) had been completed. Do not issue CHECK for this RPL.

Registers Upon Entry: When the DFASY exit routine receives control, register 1 contains the address of a 5-word parameter list (the parameter list is summarized in Figure 3):

- The first word contains the address of an ACB. This ACB is the ACB of the application program to which the input data was sent.
- The second word contains the CID of the terminal that sent the data.
- The third word contains whatever has been placed in the USERFLD field of the NIB associated with that terminal.
- The fourth word is reserved.
- The fifth word contains the address of a VTAM-supplied read-only RPL. Other than the fact that it resides in read-only VTAM storage and cannot be used by an RPL-based macro instruction, the read-only RPL is identical to any other RPL. The application program can examine the read-only RPL fields with SHOWCB and TESTCB macro instructions or with assembler instructions. The read-only RPL feedback fields are set exactly as they would be following a RECEIVE (RTYPE=DFASY) macro instruction except that the REQ field is not set.

Other general purpose registers contain the following:

Register 14: The address in VTAM to which the DFASY routine must branch when it has finished processing. VTAM will handle the return of control to the instruction in the application program that was about to be executed when the DFASY interruption occurred.

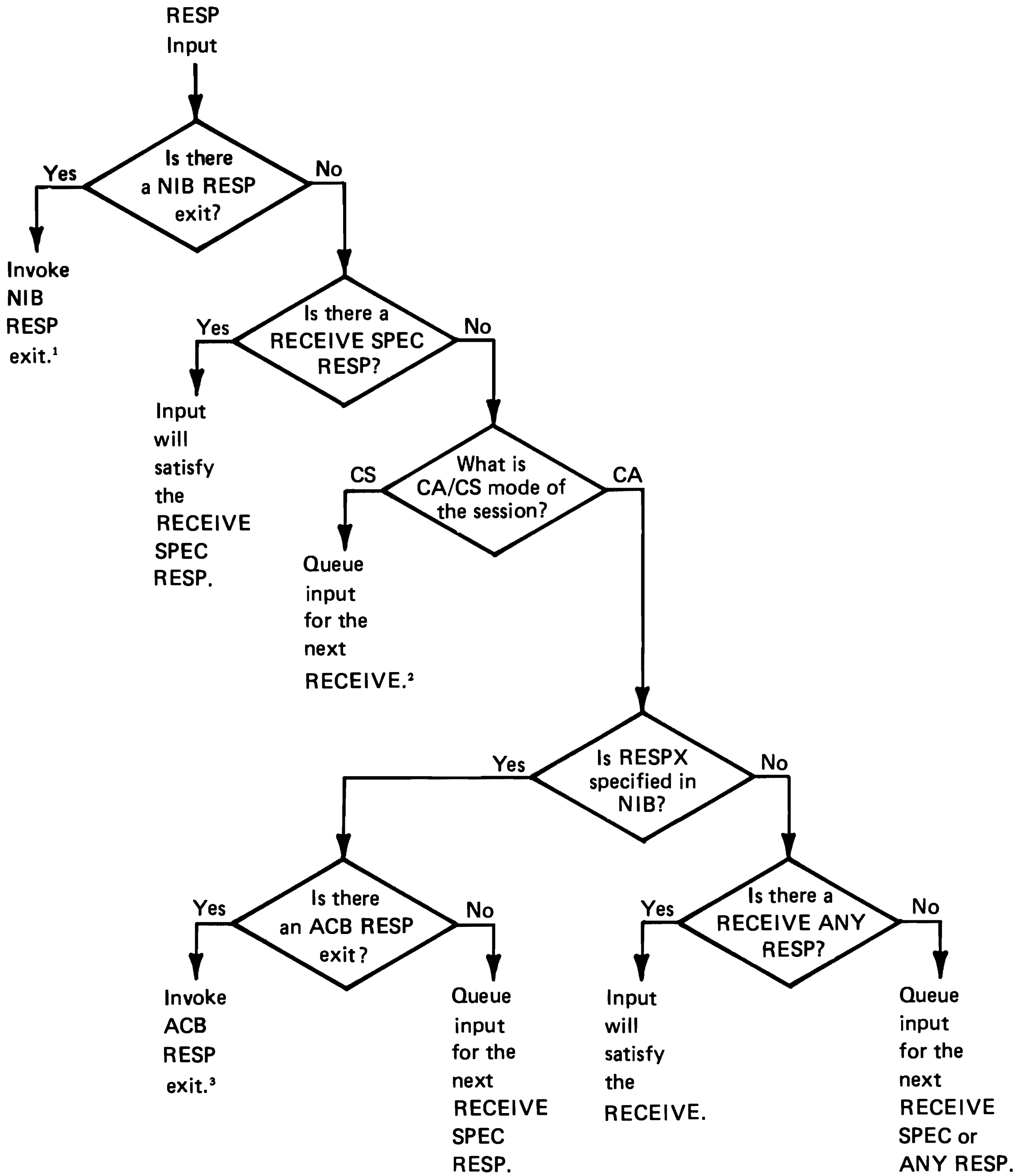
Register 15: The address of the DFASY routine.

The contents of the remaining registers (0 and 2-13) are unpredictable.

RESP=exit routine address (Record mode only)

The EXLST containing the RESP exit routine address can be pointed to by a NIB, as well as by an ACB (see the EXLST operand of the NIB macro instruction).

The RESP operand indicates the address of a routine to be entered when responses to messages arrive from a logical unit. VTAM handles the response in this manner:



Notes: [1] The exit routine is scheduled if no other exit routine (including the NIB RESP exit routine) is currently running; if there is an exit routine running, the input is queued for the NIB RESP exit routine.

[2] The input will satisfy a RECEIVE SPEC RESP. The input can also be obtained by a RECEIVE ANY RESP if the receive mode has been switched to CA.

[3] The exit routine is scheduled if no other exit routine (including the ACB RESP exit routine) is currently running; if there is an exit routine running, the input is queued for the ACB RESP exit routine.

The RESP exit routine provides a way for VTAM to notify the application program that a response to a message has arrived. The application program could maintain an active RECEIVE macro instruction for this purpose, but the RECEIVE macro instruction requires that an active RPL be committed before the response arrives, while the RESP exit routine does not.

No RECEIVE is issued in the RESP exit routine to receive the response. Instead, the exit routine is passed the address of a read-only RPL. The read-only RPL fields are set as though a RECEIVE macro instruction (RTYPE=RESP) has been completed. These fields can be examined with SHOWCB and TESTCB macro instructions or with assembler instructions like any other RPL. Do not issue CHECK for this RPL.

Registers Upon Entry: When the RESP exit routine receives control, the register contents are the same as those described above for the DFASY exit routine. That is:

Register 1: The address of a parameter list containing the ACB address, the logical unit's CID and USERFLD data, and the address of the read-only RPL. The parameter list is summarized in Figure 3.

Register 14: The address in VTAM to which the RESP exit routine must return when it is finished processing. VTAM returns control to the instruction in the application program that was about to be executed when the RESP interruption occurred.

Register 15: The address of the RESP exit routine.

The contents of the remaining registers (0 and 2-13) are unpredictable.

SCIP=exit routine address (Record mode only)

The EXLST containing the SCIP exit routine address can be pointed to by a NIB, as well as by an ACB (see the EXLST operand of the NIB macro instruction).

The SCIP operand indicates the address of a routine to be scheduled when a Request Recovery (RQR) command arrives from a logical unit. (The SCIP exit routine is the only way the application program can be notified of the arrival of an RQR command.) VTAM automatically responds to the RQR.

If an SCIP exit routine is specified in an ACB (using the EXLST macro instruction) and if it is available, it is invoked unless there is an available SCIP exit routine specified in the NIB. In that case, the SCIP exit routine specified in the NIB is invoked.

No RECEIVE macro instruction is used in the SCIP exit routine. The address of a read-only RPL is provided in the SCIP parameter list. The application program should test the CONTROL field of the read-only RPL to verify that an RQR command has arrived (CONTROL=RQR for a TESTCB macro instruction).

An RQR command may signify that the logical unit has discovered a discrepancy between the sequence numbers of its messages and the sequence numbers of the corresponding responses to those messages. The logical unit is in effect asking the application program to stop all message and response flow, establish the correct sequence number, and resume message and response flow. These actions are accomplished by issuing Clear, Set and Test Sequence Number (STSN), and Start Data Traffic (SDT) commands with a SESSIONC macro instruction. (The logical unit cannot accomplish this recovery procedure itself because only the application program can issue Clear, STSN, and SDT commands.)

Registers Upon Entry: When the SCIP exit routine receives control, the register contents are the same as those described above for the DFASY exit routine. That is:

Register 1: The address of a parameter list containing the ACB address, the logical unit's CID and USERFLD data, and the address of the read-only RPL. The parameter list is summarized in Figure 3.

Register 14: The address in VTAM to which the SCIP exit routine must return.

Register 15: The address of the SCIP exit routine.

The contents of the remaining registers (0 and 2-13) are unpredictable.

TPEND=exit routine address

Indicates the address of a routine to be entered when the network operator issues a HALT command, when VTAM detects an internal problem that necessitates halting itself, or when VTAM abnormally terminates.

If the network operator issues a HALT command to close the network normally, I/O or connection requests that are being processed are completed before closedown. Pending connection requests cause the LOGON exit routine, if available, to be scheduled; attempts to accept the connection, however, will fail. Therefore, the LOGON exit routine should be notified that the TPEND exit routine has been entered. The LOGON exit routine can thus avoid unnecessary processing of connection requests. New requests are not permitted. The TPEND exit routine, in response to a normal HALT command, should issue an asynchronous CLSDST macro instruction for each connected terminal or cause the main program to issue a CLOSE macro instruction, which will result in the eventual disconnection of all the terminals.

If the network operator issues a HALT QUICK command to close the network or if VTAM detects an internal problem that necessitates halting itself, I/O or connection requests that are being processed are canceled, and the FDBK2 field of the RPL is set to indicate the reason for the cancelation. New requests are not permitted. SYNAD and LOSTERM exit routines, if available, are scheduled for any terminals that are connected at the time TPEND is entered. The TPEND exit routine, in response to a HALT QUICK command or VTAM halting itself, should not attempt to communicate with any of the connected terminals. It should issue an asynchronous CLSDST macro instruction for each connected terminal or cause the main program to issue a CLOSE macro instruction, which will result in the eventual disconnection of all the terminals.

In an OS/VS1 or OS/VS2 MVS system, if VTAM abnormally terminates or if the network operator issues a HALT CANCEL command (which also causes VTAM to abnormally terminate) I/O and connection requests that are being processed are canceled. The FDBK2 field of the RPL is *not* set to indicate the reason for the cancelation. New requests are not permitted. SYNAD and LOSTERM exit routines are not scheduled. The TPEND exit routine, in response to a HALT command canceling VTAM or to VTAM abnormally terminating, should not attempt to communicate with any of the connected terminals. It should cause the main program to issue a CLOSE macro instruction, which will result in the disconnection of all the terminals at once; it may not issue CLSDST macro instructions. In DOS/VS and OS/VS2 SVS the HALT command to cancel VTAM is not recognized, and VTAM abnormal termination is handled exactly as a HALT QUICK command.

Note: *A CLOSE macro instruction cannot be issued in an exit routine, but the TPEND exit routine can cause a CLOSE macro instruction to be issued in the main program (by posting an ECB upon which the main program is waiting, for example).*

If there is no TPEND exit routine, the VTAM application program is not notified that a normal or quick HALT has been issued or that VTAM is halting itself except by means of the FDBK2 field of the RPL after a pending or new request has failed. In addition, the application program is abnormally terminated when a HALT command canceling VTAM is issued or when VTAM abnormally terminates. In all of these situations, it is imperative that the terminals be disconnected. Failure to do so may result in the termination of the application program by the network operator. It is therefore strongly recommended that the application program include a TPEND exit routine.

Registers Upon Entry: When the TPEND exit routine receives control, register 1 contains the address of a 2-word parameter list (the parameter list is summarized in Figure 3):

- The first word contains the address of an ACB. This ACB is the ACB of the application program being shut down.
- The value of the second word indicates the reason for the shutdown:

  0 The network operator issued a HALT command to close the network normally.

  4 The network operator issued a HALT command to close the network quickly, or VTAM detected an internal problem and is halting itself.

  8 The network operator issued a HALT command to cancel and abnormally terminate VTAM or VTAM has abnormally terminated (OS/VS1 and OS/VS2 MVS only).

Other general purpose registers contain the following:

Register 14: The address in VTAM to which the TPEND exit routine must branch when it is through processing. VTAM returns control to the instruction in the application program that was to be executed when the TPEND interruption occurred.

Register 15: The address of the TPEND exit routine.

The contents of the remaining registers (0 and 2-13) are unpredictable.

RELREQ=exit routine address

Indicates the address of a routine that is entered when another application program (or TOLTEP) requests connection to a terminal that is currently connected to your application program. This occurs when the other application program issues a SIMLOGON macro instruction (with the RELRQ and Q options specified) on behalf of your terminal.

The RELREQ exit routine may want to determine whether there are any I/O requests pending for the terminal and release it only after these I/O operations have been completed. If the application program wants to release the terminal (this is optional), it should disconnect the terminal with a CLSDST macro instruction. This CLSDST macro instruction must have the RELEASE option code in effect for its RPL. The terminal is not disconnected until CLSDST is executed. When CLSDST is executed, the terminal is made available to the application program that has been waiting for it the longest amount of time.

If you have no RELREQ exit routine, your application program cannot be notified at the other application program's request. If the request was issued with NQ set, the request is rejected. If Q was set, the request remains pending until you disconnect the terminal with CLSDST.

The application program that causes your RELREQ exit routine to be invoked may have had the CONANY option code in effect for its SIMLOGON request. Although the use of CONANY causes VTAM to ultimately connect only *one* terminal, VTAM may in the process invoke *many* RELREQ routines. Thus you may release your terminal, only to have it remain unconnected when the other application program's request is satisfied by some other terminal. To prevent this problem, follow the CLSDST with an OPNDST (OPTCD=ACQUIRE) or SIMLOGON request of your own. Then if the other application program is ignoring your just-released terminal, you get it back.

Registers Upon Entry: When the RELREQ exit routine receives control, register 1 contains the address of a 2-word parameter list (the parameter list is summarized in Figure 3):

- The first word of the parameter list contains the address of an ACB. This ACB is the ACB through which the terminal is currently connected to an application program.
- The second word of the parameter list contains the address of the symbolic name of the requested terminal. The name is 8 bytes long and padded on the right with blanks, if necessary.

The other registers contain the following:

Register 14: The address in VTAM to which the RELREQ routine must branch when it is through processing. VTAM will handle the return of control to the instruction in the application program that was about to be executed when the RELREQ interruption occurred.

Register 15: The address of the RELREQ routine.

The contents of the remaining registers (0 and 2-13) are unpredictable.

LOGON=exit routine address

Indicates the address of a routine to be entered when VTAM has queued a logon for the application program.

VTAM queues a logon if (1) the logical unit has issued an Initiate Self command or a character-coded logon, (2) a terminal issues a logon via the network solicitor, (3) the application program to which the terminal is currently connected issues a CLSDST macro instruction with OPTCD=PASS, (4) an application program issues a SIMLOGON macro instruction on behalf of the terminal, or (5) the user has specified automatic logons for the application program from the terminal when the network was defined or the network operator has specified an automatic logon using the VARY command. These cause the LOGON exit routine to be scheduled if SETLOGON (OPTCD=START) is in effect.

Note: *When a terminal logs on, VTAM first checks for outstanding OPNDST requests (that is, OPNDST= with ACCEPT and Q that have not yet been completed because no logon request has been made by the terminal operator). If there are no outstanding OPNDST requests, VTAM checks for a LOGON exit routine, and schedules the routine if an active one is found. Thus a logon will not cause a LOGON exit routine to be scheduled if there is a pending OPNDST with ACCEPT. If no LOGON exit routine exists, the logon is queued.*

For automatic logons: If a terminal has been defined by the user as a dial-in terminal (CALL=IN specified in the LINE or GROUP NCP definition macro), the LOGON exit routine is scheduled when the terminal dials in. If a terminal has been defined by the user as a dial-out or in/out terminal (CALL=OUT or INOUT), the LOGON exit routine is scheduled when the application program opens its ACB and issues SETLOGON with OPTCD=START. The terminal will not be dialed until OPNDST is completed (for logical units) or until the first I/O request is directed to it (for BSC and start-stop terminals).

Regardless of the mechanism by which the LOGON exit routine is scheduled, the routine is in effect being asked to connect the terminal to the application program. The routine's principal task therefore is to determine if it should honor the request and, when it determines that it should, issue an OPNDST macro instruction to establish connection with the terminal. If the request is not to be honored, the routine should issue the CLSDST macro instruction for the terminal (which removes the terminal from the logon queue). If neither OPNDST nor CLSDST is issued, the terminal may remain unconnected to any application program.

If the NIB specifies MACRF=LOGON and SETLOGON (OPTCD=QUIESCE) has not been issued, logons are queued for your application program regardless of whether a LOGON exit routine is available. A logon remains queued until you issue OPNDST or CLSDST for the terminal.

Note that the "queuing" of a logon does not necessarily mean that the logon is queued for eventual scheduling of the LOGON exit routine; it merely means that the logon is queued for an eventual OPNDST (OPTCD=ACCEPT) macro instruction (or CLSDST).

The LOGON exit routine can issue an INQUIRE macro instruction to obtain the session parameters determind by the logon mode name and the logon message supplied by the terminal that is logging on. If the routine determines from the session parameters and the logon message that connection with the terminal is acceptable, it may wish to establish that connection. This is accomplished by using information passed to the LOGON exit routine, along with information obtained with the INQUIRE macro instruction, to build or modify a NIB and an RPL, and by then issuing the OPNDST macro instruction with ACCEPT and SPEC option codes.

The LOGON exit routine is entered only if MACRF=LOGON was specified for the ACB, and the application program has issued the SETLOGON (OPTCD=START) macro instruction.

Registers Upon Entry: When the LOGON exit routine receives control, register 1 contains the address of a 4-word parameter list (the parameter list is summarized in Figure 3):

- The first word contains the address of an ACB. This ACB is the ACB to which the logon request was directed. The ACB address should be specified for the ACB operand of an INQUIRE macro instruction used to obtain the data portion of the logon.
- The second word contains the address of the 8-byte symbolic name of the terminal requesting logon. This name should be placed in the NAME field of the NIB used to establish connection with the terminal. (The symbolic name being pointed to here is the same as the name of the terminal's entry in the resource definition table. The logical unit's entry is an LU entry. The terminal's entry is either a TERMINAL or COMP entry, or, if the terminal is a start-stop or BSC dial-in terminal without an automatic ID verification feature, the UTERM name

in a TERMINAL entry. LU, TERMINAL, and COMP are VTAM definition statements used by the user to build entries in the resource definition table.)

- The third word is reserved.
- The fourth word contains the length of the data portion of the logon sent by the terminal. This length should be used with the LENGTH operand of any INQUIRE macro instruction used to obtain the data portion of the logon.

Other registers contain the following:

Register 14: The address in VTAM to which the LOGON exit routine should branch when it is through processing. VTAM handles the return of control to the application program instruction that was about to be executed when the LOGON interruption occurred.

Register 15: The address of the LOGON exit routine.

The contents of the remaining registers (0 and 2-13) are unpredictable.

LOSTERM=exit routine address

Indicates the address of a routine to be entered when contact with a terminal has been lost, when the logical unit has requested a logoff, when certain errors are detected in transmission, or when the terminal is temporarily unavailable. As noted below, the application program may or may not issue CLSDST to disconnect the terminal. (If the application program fails to issue CLSDST, the terminal will remain unavailable for connection to any other application program.)

If there is no LOSTERM exit routine, the application program is not notified that the terminal is unavailable, except via return codes following pending and subsequent I/O requests for the terminal. A LOSTERM exit routine is especially recommended for those application programs that do not issue specific-mode I/O requests for their terminals, but are instead driven by input arriving as the result of READ or RECEIVE macro instructions issued in the any-mode. It is also recommended for application programs that issue specific mode I/O requests when there is the possibility that the terminal may fill VTAM's buffers faster than the application program is emptying them with RECEIVE macro instructions.

Because the LOSTERM exit routine is scheduled asynchronously, the LOSTERM exit routine must be aware of the current status of the device (terminal or logical unit) for which the exit routine was scheduled. An application program can keep status indicators for each device to which it plans to be connected. For more information, refer to *VTAM Macro Language Guide*.

Registers Upon Entry: When the LOSTERM exit routine receives control, register 1 contains the address of a 4-word parameter list (the parameter list is summarized in Figure 3).

- The first word contains the address of an ACB. This ACB is the ACB of the application program to which the terminal was connected.
- The second word contains the CID of the terminal. The ARG field of an RPL used for CLSDST must contain this CID.
- The third word contains whatever had been placed in the USERFLD field of the NIB associated with the terminal.
- The value contained in the fourth word indicates why the LOSTERM exit routine was entered:

Value (in decimal)	Meaning
0	A dial-line disconnection occurred for a dial-in BSC or start-stop terminal. A CLSDST macro instruction is required.
4	A dial-line disconnection occurred for a dial-out BSC or start-stop terminal. If no data from the terminal remains in VTAM buffers, a READ or WRITE (OPTCD=SPEC) macro instruction will redial the terminal. If redialing fails (causing the LOSTERM exit routine to be rescheduled) the CLSDST macro instruction should be issued for the terminal.
8	Reserved.
12	Contact with a logical unit (on either a switched or non-switched line) or a BSC, or a start-stop terminal (on a non-switched line) was permanently lost for one of the following reasons: (1) the network operator has issued a VARY command for the terminal. (For VARY INACT,F, the LOSTERM exit routine will be entered twice with a return code of 12), (2) the communication controller's NCP has begun an automatic network shutdown or has abended and cannot be restarted, (3) there has been a permanent channel failure between the CPU and the communication controller or locally attached terminal, (4) there has been a failure in the network path between the communication controller and the remotely attached terminal, (5) the network operator issues a HALT NET, QUICK command, or (6) a test request message has been received from the terminal other than a 3270 in basic mode (see *DOS/VS and OS/VS TOLTEP for VTAM*, GC28-0663, for more information about test request messages). For logical units, VTAM automatically issues a Clear command for the application program. For BSC and start-stop terminals, READ macro instructions may be issued to obtain data already sent from the terminal. Unless a CLSDST has already been executed successfully, one must be issued if the application program is still in session with the logical unit.
16	The logical unit has been successfully recontacted. VTAM has automatically issued a Clear command for the application program. If the application program is still in session with the logical unit, issue a CLSDST macro instruction. If the logical unit was acquired by the application program, you can issue an OPNDST or SIMLOGON macro instruction to re-acquire the logical unit. If the logical unit was accepted as the result of a pending logon, the application program may take any action it wants to reinstate the connection. **Note:** *Once the CLSDST macro instruction has been issued, reconnection of the terminal is subject to the normal rules for acquisition. Therefore, if another application program has a connection request queued for the terminal, or the terminal has issued a connection request for another application program, the terminal may not be immediately reconnected to the releasing application program.*
20	An unconditional Terminate Self or an unconditional character-coded logoff command was issued by the logical unit. VTAM automatically issues a Clear command for the application program. A CLSDST macro instruction is required.

Value (in decimal)	Meaning
24	Contact with the logical unit has been lost but VTAM may be able to reestablish it. Stop output to the logical unit and either return to VTAM or issue a CLSDST macro instruction. If you do not issue a CLSDST, you must return to VTAM; VTAM will attempt to recontact the logical unit. If it is successful and the application program is still in session with the logical unit, VTAM reschedules the LOSTERM exit routine with a return code of 16; if it is unsuccessful and the application program is still in session with the logical unit, the LOSTERM exit routine is rescheduled with a return code of 12. If you issue a CLSDST for the logical unit, the LOSTERM exit routine will not be rescheduled and you will not get return code 12 or 16. **Note:** *If you intend to resume communication if the logical unit is successfully recontacted, you can issue RECEIVE (OPTCD=NQ) macro instructions to obtain any data still in the network. (If data is still in the network when the logical unit is recontacted, the data is discarded.) These RECEIVEs must be issued before the CLSDST is issued.*
28	Incomplete or invalid segmented requests were received from a logical unit that can transmit segmented requests. Issue a SESSIONC (CONTROL=CLEAR) and SESSIONC (CONTROL=SDT) when appropriate for the transmission services profile specified by the session parameters (see Appendix J) if the session is to be continued, or issue a CLSDST macro.
32	A conditional Terminate Self command or a conditional character-coded logoff has been issued by the logical unit. The application program may take any action it desires including issuing a CLSDST for the logical unit.
36	The buffer limit defined for a logical unit has been exceeded. VTAM automatically issues a Clear command for the application program. Any data for which the application program has not issued a RECEIVE will be discarded. The application program may resynchronize sequence numbers, or the data may be retransmitted after issuing SESSIONC (CONTROL=SDT) if appropriate for the transmission services profile specified by the session parameters. (See Appendix J.)
40	The operator at a BSC or local non-SNA 3270 terminal in basic mode has hit the Test Request Key and the terminal is being taken over by TOLTEP. A CLSDST macro instruction is required. **Note:** *For any of the LOSTERM return codes that require or recommend a CLSDST macro instruction, do not issue a second CLSDST if one has already been issued to the same terminal but for a different reason.*

Other general purpose registers contain the following:

Register 14: The address in VTAM to which the LOSTERM exit routine must branch when it is through processing. VTAM handles the return of control to the point in the application program where the LOSTERM interruption occurred.

Register 15: The address of the LOSTERM exit routine.

The contents of the remaining registers (0 and 2-13) are unpredictable.

ATTN=exit routine address (Basic mode only)

Indicates the address of a routine to be entered when a start-stop terminal connected to the application program causes an attention interruption and no read or write operation is pending or in progress for the terminal. (See Figure 4.)

Such an attention interruption causes an error lock to be set for the terminal by the CPU or the communications controller; no I/O can be performed with the terminal until this error lock is reset with a RESET macro instruction.

If there is no ATTN exit routine to be invoked, the attention interruption is ignored, and the error lock is automatically reset.

The ATTN exit is taken only if (1) the application program specified PROC=MONITOR in the NIB representing the terminal that issued the attention interruption and (2) the terminal is a 2741, 1050, Communicating Magnetic Card Selectric Typewriter, or an AT&T Teletypewriter Terminal.

The application program is notified of attention interruptions that occur *during* an I/O operation by means of a return code set in the FDBK2 field of the I/O request's RPL.

Registers Upon Entry: When the ATTN routine receives control, register 1 contains the address of a 3-word parameter list (the parameter list is summarized in Figure 3):

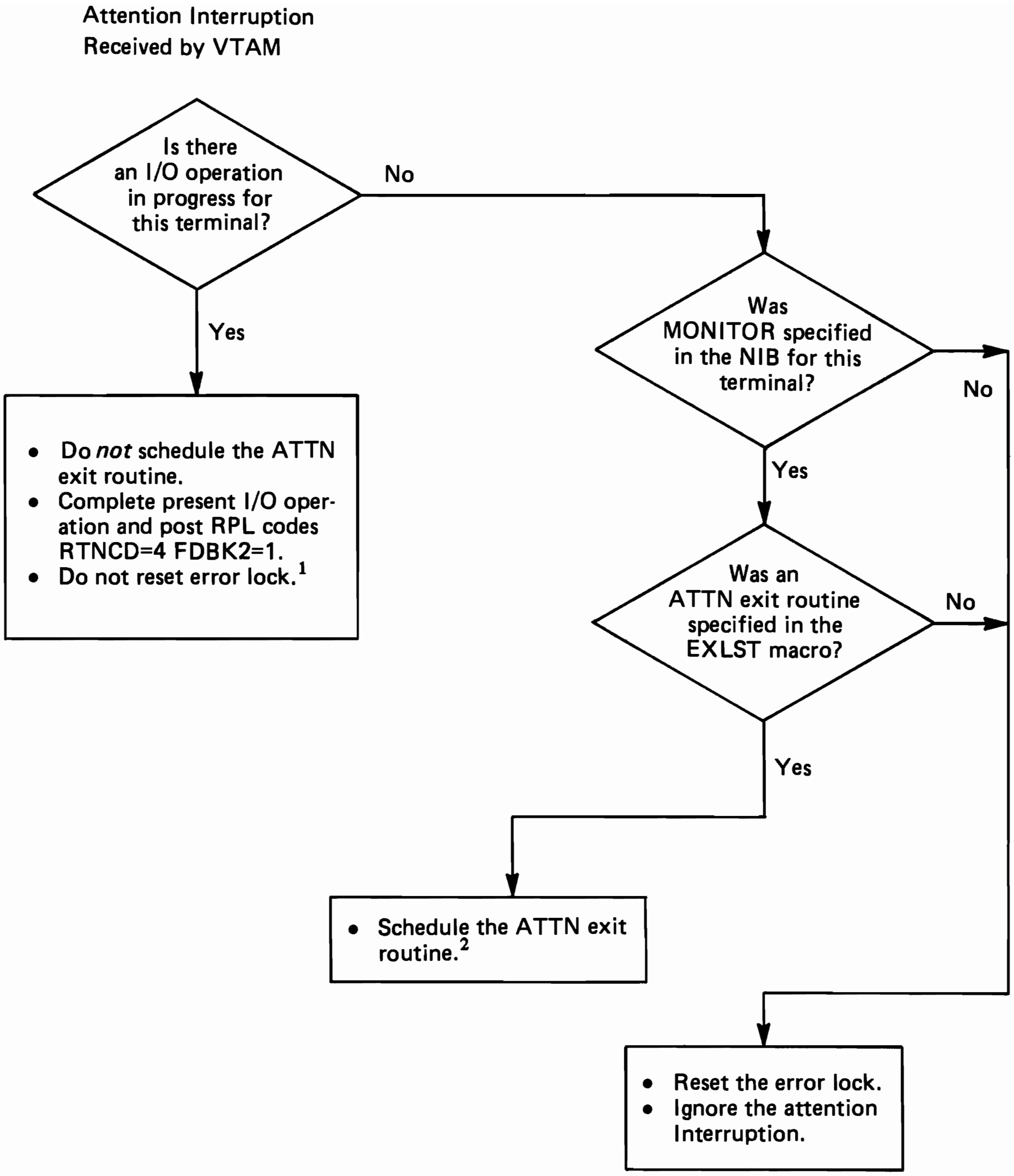
- The first word contains the address of an ACB. This ACB is the ACB through which the terminal issuing the attention interruption is currently connected.
- The second word contains the CID of the terminal. The ARG field of any RPL used to communicate with this terminal must contain this CID.
- The third word contains whatever was placed in the USERFLD field of the NIB associated with the terminal.

Other general purpose registers contain the following:

Register 14: The address in VTAM to which the ATTN exit routine should branch when it is through processing. VTAM handles the return of control to the application program at the point that the attention interruption occurred.

Register 15: The address of the ATTN exit routine.

The contents of the remaining registers (0 and 2-13) are unpredictable.



Notes: [1] Until this error lock is reset (with a RESET macro instruction), all subsequent I/O operations will fail with RTNCD=X'0C' FDBK2=X'00'.

[2] The application program's ATTN exit routine must determine whether to cancel any pending I/O requests and reset the error lock.

Figure 4. How VTAM Handles Attention Interruptions

GENCB–Generate a Control Block

The GENCB macro instruction builds an ACB, EXLST, RPL, or NIB. The advantage of using the GENCB macro instruction is that the control blocks are generated during program execution. (With the ACB, EXLST, RPL, and NIB macro instructions, the control blocks are built during program assembly.) If GENCB, MODCB, TESTCB, and SHOWCB are used to build and manipulate the control blocks, program reassembly should not be required should control block formats be changed during future releases of VTAM.

GENCB not only builds the control block during program execution, but can also build the control block in dynamically allocated storage. One advantage of this technique is that it can remove application program dependencies on the length of each control block.

The GENCB user specifies the type of control block to be built and the contents of its fields. The operands used to specify the field contents are exactly the same as those used in the macro instruction that builds the control block during assembly. For example, these macro instructions build the same exit list:

```
GENCB     BLK=EXLST,SYNAD=SYNADPGM,AM=VTAM
EXLST     SYNAD=SYNADPGM,AM=VTAM
```

The control block is built either in storage that VTAM obtains via the OS/VS GETMAIN or DOS/VS GETVIS facility, or in the application program's storage. To accomplish the latter, the application program should either reserve enough storage during program assembly to accomodate the control block, or perform its own GETMAIN or GETVIS operation to obtain the necessary storage. If the application program is providing the storage, the location and length of this storage must be coded in the GENCB macro instruction. Dynamic storage allocation for the control block occurs automatically if the location and length operands (WAREA and LENGTH) are omitted. The application program can issue FREEMAIN or FREEVIS macro instructions to free the storage obtained by GENCB. (If FREEMAIN is used, return the storage to subpool 0. If GENCB is issued in a task running in privileged state, return the storage to subpool 252.)

Dynamic storage allocation can be successful only if (1) the program is operating in virtual mode and (2) enough unallocated virtual storage remains in the program's partition or region to build the control block. See the description of the LENGTH operand for an explanation of how control block lengths are determined.

List, generate, and execute forms of the GENCB macro instruction are available; they are designated by the MF operand. These forms must be used in reenterant routines such as the LERAD and SYNAD exit routines (see the VTAM *Macro Language Guide*.

Because there is a large variety of formats in which the GENCB operands can be specified, format specifications have been tabulated in Appendix E and do not appear in this macro instruction description.

Name	*Operation*	*Operands*
[symbol]	**GENCB**	**BLK= {ACB | EXLST | RPL | NIB } ,AM=VTAM [, keyword=value]... [, COPIES=quantity] [,WAREA=work area address, LENGTH=work area length] [, MF=list, generate, or execute form parameters]**

BLK=ACB | EXLST | RPL | NIB

Indicates the type of control block to be generated.

AM=VTAM

Identifies this macro instruction as a VTAM macro instruction. This operand is required.

keyword=value

Indicates a control block field and the value that is to be contained or represented within it.

For *keyword*, code any keyword that can be used in the macro instruction corresponding to the BLK operand. If BLK=ACB is used, for example, code the keyword of any operand that can be used in the ACB macro instruction. One exception: ARG=(register) can also be coded if BLK=RPL.

For *value*, indicate a register or code any value that could be used if the operand were being specified in the ACB, EXLST, RPL, or NIB macro instruction, or use one of the formats indicated in Appendix E.

Note: *If no keywords are included, the following types of control blocks are built:*

ACB: All fields are set to 0, and the MACRF field is set to NLOGON.

RPL: All fields are set to their default values (as indicated in the RPL macro instruction description).

EXLST: All fields are set to 0.

NIB: All fields are set to their default values (as indicated in the NIB macro instruction description).

COPIES=quantity

Indicates the number of control blocks to be generated.

The copies are identical in form and content. With the exception of an exit list, they are placed contiguously in storage, whether that storage is the area indicated by the WAREA operand or is dynamically allocated storage. Exit lists begin on the next fullword boundary following the end of the previous exit list.

The length returned in register 0 is the total length of the generated control blocks. The length of each block (the total length divided by the number of copies) can be used to determine the location of the beginning of each block.

Note: *If this operand is not used, one control block is built.*

WAREA=work area address

Indicates the location of the storage area in the application program where the control block is to be built. The work area must be aligned on a fullword boundary. If this operand is specified, the LENGTH operand must also be specified.

If the WAREA and LENGTH operands are omitted, VTAM obtains dynamically allocated storage via the GETMAIN or GETVIS facility and builds the control block there. Assuming that GENCB is completed successfully (this is indicated by a return code of 0 in register 15), the address of the generated control block (or blocks) is placed in register 1, and their total length is placed in register 0.

LENGTH=work area length

Indicates the length (in bytes) of the storage area designated by the WAREA operand.

If this length is insufficient, register 15 will contain the value 4, and register 0 will contain the value 9.

To avoid having to recode your application program should you wish to run it under a different operating system, use the manipulative macro instructions to obtain the control block lengths. You do this by specifying ACBLEN, EXLLEN, RPLLEN, or NIBLEN in either a SHOWCB or TESTCB macro instruction. For example, to obtain the length of an ACB in your particular operating system, the following SHOWCB could be coded:

```
SHOWCB    FIELDS=ACBLEN,AREA=WORKAREA,LENGTH=4,AM=VTAM
```

Or, to test the length of an exit list in your particular operating system, the following TESTCB could be coded:

```
TESTCB    EXLLEN=(7),AM=VTAM
```

If you are generating more than one control block, remember that the total length of *each* control block is the length indicated by the control block's length field (ACBLEN, EXLLEN, RPLLEN, NIBLEN) plus the number of bytes required for fullword alignment. (EXLSTs are variable in length; when no specific EXLST is specified, the length returned by SHOWCB or tested by TESTCB is the maximum possible length for your operating system.)

MF=list, generate, or execute form parameters

Indicates that a list, generate, or execute form of GENCB is to be used. Omitting this operand causes the standard form of GENCB to be used. See Appendix F for a description of the nonstandard forms of GENCB.

Examples

```
GEN1        GENCB     AM=VTAM,BLK=ACB,
                      APPLID=(3),EXLST=(6),
                      WAREA=BLOKPOOL,LENGTH=(4)
              .
              .
              .
BLOKPOOL    DS        32D
```

GEN1 builds an ACB in statically reserved storage (BLOKPOOL). When GEN1 is executed, register 3 must contain the address of an application program's symbolic name, and register 6 must contain the address of the exit list to be pointed to by the ACB.

```
            L          10,WORKAREA    (REG10=ACB LENGTH)
            GETMAIN    R,LV=(10)
            LR         5,1            (REG5=ACB ADDRESS)
GEN2        GENCB      AM=VTAM,BLK=ACB,
                       WAREA=(5),LENGTH=(10)
```

In this example, the application program is building an ACB in dynamically allocated storage obtained by itself. Using the procedure described above in the LENGTH operand description, the application program has obtained the length of an ACB and placed it in a fullword called WORKAREA. The instructions preceding GEN2 obtain the correct amount of storage, and GEN2 builds the ACB in that storage.

```
GEN3        GENCB      BLK=RPL,COPIES=10,AM=VTAM
```

GEN3 creates 10 RPLs in dynamically allocated storage obtained by VSAM (OS/VS) or VTAM (DOS/VS). The address of the beginning of these RPLs is returned in register 1, and the total length is returned in register 0. This length includes all padding for fullword alignment; the RPLLEN field indicates the length of each unpadded RPL. Each RPL is built as though an RPL macro instruction with no operands had been issued.

Return of Status Information

After GENCB processing is finished and control is returned to the application program, register 15 indicates whether or not the operation was completed successfully. If the operation was completed successfully, register 15 is set to 0; if it was completed unsuccessfully, register 15 is set to either 4, 8, or 12 (DOS/VS only). If it is set to 4 or 12, register 0 is also set indicating the specific nature of the error (see Appendix D).

INQUIRE–Obtain Terminal Information or Application Program Status

There are nine types of INQUIRE. The setting of the RPL's option code determines which one is used. The following descriptions indicate the purpose and use of these options; see the operand descriptions for details regarding how each is specified.

LOGONMSG: INQUIRE obtains the data portion of a logon from a terminal that logged on to the application program.

Note: *The data portion of a logon can only be obtained using INQUIRE prior to issuing OPNDST and as long as CLSDST has not been issued.*

DEVCHAR: INQUIRE obtains the device characteristics of a terminal, as they are defined by the user in the resource definition table. These device characteristics can be used to define which processing options the program wants to be in effect for the NIB used to connect that terminal. For a logical unit, the only indication that is returned is that of logical unit; the specific type of logical unit is not identified. This type of INQUIRE is also appropriate for use in LOGON exit routines where the program is establishing connection with terminals whose identities are not known during program assembly.

TERMS: For a given PU, LU, TERMINAL, LINE, or CLUSTER entry in the resource definition table, INQUIRE builds a NIB or list of NIBs in the application program.

The purpose of this type of INQUIRE is this: During VTAM definition, the user can define a PU, LU, TERMINAL, LINE, or CLUSTER entry and associate a set of terminals with that entry. If the application program builds one NIB that indicates this entry in its NAME field, it can then issue INQUIRE to generate NIBs for *all* of the terminals associated with the entry. Thus the application program need not be aware of the identities or the number of these terminals before establishing connection with them. This allows the user, via the network operator or VTAM definition procedures, to vary the set of terminals after the application program has been assembled.

COUNTS: For the ACB specified in the RPL, INQUIRE provides the number of terminals that are currently connected and the number of terminals that have entered a logon or have had a logon entered for them but have not yet been connected.

APPSTAT: INQUIRE checks the status of a specified application program and determines whether the application program is accepting logons, never accepts logons, is temporarily not accepting logons, not longer accepts logons, or has not yet opened its ACB. A code representing each situation is returned in the RPL's FDBK field.

CIDXLATE: Given a terminal's CID, INQUIRE provides the symbolic name of that terminal. Conversely, given the symbolic name of a connected terminal, INQUIRE provides the corresponding CID of that terminal.

When a terminal is connected to an application program, the symbolic name of that terminal is converted into a 4-byte equivalent called the CID. This CID must subsequently be used for all I/O requests for the terminal.

TOPLOGON: When a terminal directs a logon at an application program (ACB) or when a logon is made on its behalf, the application program may or may not immediately accept or reject the terminal. Until the logon is accepted or rejected, it is said to be *queued* to the ACB. More than one can be queued to the ACB. The TOPLOGON option supplies the symbolic name of the terminal that is currently at the head of the logon queue for a given ACB.

BSCID: This version of INQUIRE is used when a BSC terminal with an ID verification feature dials in and causes a logon to be generated for the application program. If the application program determines that the terminal's name is one that was associated with an IDLST having NOMATCH=PASS in effect (see your system programmer), INQUIRE with OPTCD=BSCID supplies the terminal's ID verification sequence.

SESSPARM: Depending upon the parameters specified for the LOGMODE operand of the NIB, INQUIRE obtains the session parameters that are associated with a pending logon from a logical unit or obtains the session parameters associated with an entry in the logon mode table defined for the logical unit named in the NIB.

Name	*Operation*	*Operands*
[symbol]	**INQUIRE**	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address
Indicates the location of the RPL that indicates which kind of processing INQUIRE is to perform.

rpl field name=new value
Indicates an RPL field to be modified and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the INQUIRE macro instruction.

Format: For *rpl field name,* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register. ARG=(register) can also be specified.

The following RPL operands apply to the INQUIRE macro instruction:

ACB=acb address
Indicates the ACB that identifies the application program.

ARG=(register)
Indicates the register containing the CID of the terminal. Register notation must be used if the CID is to be placed in the ARG field with this INQUIRE macro instruction. This operand applies to the DEVCHAR and CIDXLATE forms of INQUIRE.

NIB=nib address
Indicates the NIB whose NAME field identifies the terminal or application program. This operand applies to the LOGONMSG, DEVCHAR, TERMS, APPSTAT, BSCID, SESSPARM, and CIDXLATE forms of INQUIRE. For DEVCHAR and CIDXLATE, NIB=address and ARG=(register) are mutually exclusive methods of identifying the terminal.

AREA=address of data area
Indicates where the information produced by INQUIRE is to be placed.

AREALEN=length of data area
Indicates the maximum number of bytes of data that the data area can hold; if the data to be placed there exceeds this value, a special condition results (RTNCD=0, FDBK2=5) and the RECLEN field indicates the required length. The INQUIRE can be reissued using the correct length.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) INQUIRE macro instruction is completed. The macro instruction is completed when the information has been placed in the application program's storage area. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field of the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When the SYN option code is set, control is returned to the application program when the INQUIRE macro instruction has been completed. When ASY is set, control is returned as soon as VTAM has accepted the request. Once the INQUIRE operation has been completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field.

OPTCD=LOGONMSG | DEVCHAR | COUNTS | TERMS | APPSTAT | CIDXLATE | TOPLOGON | BSCID | SESSPARM

LOGONMSG
INQUIRE obtains a logon message from a terminal that has requested logon for the application program.

The RPL's ACB field must indicate the ACB to which the logon is directed. The NIB field must point to a NIB whose NAME field contains the symbolic name of the terminal issuing the logon. The AREA and AREALEN fields must indicate the location and length of the storage area where the logon message is to be placed.

Note: *The information required for the ACB, NAME, and AREALEN fields is passed to the LOGON exit routine in a parameter list.*

VTAM indicates the length of the logon message in the RPL's RECLEN field. If the message is too long to fit, RECLEN is posted with the required length. Conditional completion is indicated (RTNCD=0 and FDBK2=5), and no data is supplied to the application program.

Note: *The data portion of a logon cannot be obtained after an OPNDST or CLSDST is issued.*

DEVCHAR

INQUIRE obtains the device characteristics of a terminal, as they are defined by the user in the resource definition table.

The RPL must indicate the terminal in one of two ways: either the RPL's NIB field must indicate a NIB containing the symbolic name of the terminal, or the RPL's ARG field must contain the CID of the terminal.

The device characteristics are placed in an 8-byte program storage area whose location is set in the AREA field. The AREALEN field must be set to 8. The bits that are set in this area indicate whether the device is an input, output, or input/output device. The specific device type (for example, 2741 communications terminal) is also indicated, along with additional information. See the description of the ISTDVCHR DSECT in Appendix H for a complete description of the DEVCHAR information.

TERMS

For a given PU, LU, TERMINAL, LINE, CLUSTER, or GROUP entry in the resource definition table, INQUIRE builds a NIB or list of NIBs in the application program.

The RPL's NIB field must point to a NIB whose NAME field contains the name of an entry that exists in the resource definition table at the time INQUIRE is executed. A NIB is built for each terminal represented in the entry.

The AREA and AREALEN fields designate the location and length of the work area where the NIBs are built. The work area must be set to binary zeroes by the application program before INQUIRE is issued.

VTAM indicates the total length of the NIBs in the RPL's RECLEN field.

If the application program wants the NIBs to be built in dynamically allocated storage (obtained by the application program), INQUIRE should be issued twice. For the first INQUIRE, set AREALEN to 0. This INQUIRE will be completed with RTNCD=0 and FDBK2=5 (insufficient length) and RECLEN will indicate the required length. Obtain the storage and issue INQUIRE with AREALEN set to the proper length.

Each NIB contains the symbolic name of the terminal, with flags for the LISTEND field set in such a way as to group the NIBs together into a NIB list. In addition, device characteristics are placed in each NIB. These characteristics can be used to reset the PROC options of the NIB to values that are appropriate for the terminal.

After the user has set each NIB's MODE field to BASIC or RECORD and other NIB fields to their desired values, the NIBs are ready to be used for connection.

COUNTS

For the ACB specified, INQUIRE provides the number of terminals that are currently connected and the number of terminals that have entered a logon or have had a logon entered for them but have not yet been connected.

The RPL's ACB field must contain the address of the ACB. The AREA and AREALEN fields must indicate a 4-byte area where the information is to be placed. VTAM places the number of connected terminals in the first 2 bytes and the number of terminals requesting logon in the second 2 bytes.

APPSTAT

INQUIRE checks a given application program and returns one of the following values in the RPL's FDBK field:

Value (in decimal)	Meaning
0	ACTIVE: The application program is accepting logon requests (that is, the ACB is open, its MACRF field is set to LOGON, and SETLOGON START has been issued).
4	INACTIVE: The application program for this ACB is not open.
8	NEVER ACCEPTS: The application program has indicated that it never accepts logons (that is, the ACB was opened with MACRF=NLOGON specified).
12	TEMPORARILY NOT ACCEPTING: The application program has indicated that logon requests should not be directed toward it. The application program has issued SETLOGON (OPTCD=STOP) which implies that this condition is temporary, and that SETLOGON (OPTCD=START) will eventually be issued to indicate that logons can again be directed toward it. An INQUIRE issued after the application program issues SETLOGON (OPTCD=START) causes a FDBK code of 0 to be returned.
16	NO LONGER ACCEPTING: The application program has issued SETLOGON (OPTCD=QUIESCE), and logons cannot be directed toward it. Unlike a return code of 12, a return code of 16 means that the application program's logon queue is now permanently closed. Presumably, the application program is about to close its ACB.

The RPL's ACB field must contain the address of an opened ACB.

The RPL's NIB field must point to a NIB whose NAME field contains the symbolic name of the application program. (Although the NIB is generally used as a *terminal* control block, note that here it is being used to identify an *application program*. The symbolic name in the 8-byte NAME field must be left-justified and padded to the right with blanks. (This name corresponds to the application program's APPL entry in the resource definition table.)

CIDXLATE

Given a terminal's CID, INQUIRE provides the symbolic name of that terminal. Conversely, given the symbolic name of a connected terminal, INQUIRE provides the corresponding CID of that terminal.

To convert that CID back into its equivalent symbolic name, the RPL's ARG field must contain the CID when the INQUIRE macro instruction is executed. The symbolic name is returned in the data area that you indicate in the RPL's AREA field. The AREALEN field must be set to 8.

To use INQUIRE to convert the symbolic name into a CID, the RPL's NIB field must contain the address of a NIB and the named terminal must be currently connected. The NAME field of that NIB must in turn contain the symbolic name to be converted. The CID is placed in the data area that you indicate in the RPL's AREA field. The AREALEN field must be set to 4. If the terminal is not currently connected, a CID is not returned and an error code is set.

Note: *The NIB and the ARG field occupy the same physical field in the RPL. If the last macro instruction operand used to set or modify this field was ARG=(register), or if the field has been left unchanged since VTAM inserted a CID into it, VTAM recognizes that this field contains a CID. If the last operand used to set or modify this field was NIB=*address, *VTAM recognizes that the field contains a NIB address.*

TOPLOGON
INQUIRE returns the symbolic name of the terminal that has directed a logon at the application program and has spent the greatest amount of time waiting to be connected. If no logons are queued, an error return code results (see Appendix C).

The ACB field of INQUIRE's RPL must indicate the ACB whose logon queue is to be examined. The symbolic name is returned in the data area indicated by you in the RPL's AREA field. The AREALEN field must be set to 8.

BSCID
INQUIRE returns the BSC terminal's ID verification sequence. The RPL's NIB field must point to a NIB whose NAME field contains the symbolic name of the terminal (as provided in the LOGON exit routine's parameter list). The sequence, which can be up to 20 bytes long, is placed in the storage area pointed to by the AREA field. Set the AREALEN field to 20.

SESSPARM
INQUIRE returns the session parameters associated with a specified logon mode name. The NIB field of the RPL must point to a NIB whose LOGMODE operand identifies the logon mode name to be used. The logon mode name that is specified in the NIB is used to search the logon mode table defined for the logical unit named in the NIB. If a match is found, the session parameters associated with the logon mode name are returned in the AREA field of the RPL. The logon mode name may be explicitly stated in the NIB, or the NIB may indicate that the session parameters (including logon data) specified in a pending logon (character-coded logon, Initiate Self command, SIMLOGON macro, VARY LOGON operator command, or CLSDST (OPTCD=PASS) macro are to be returned. See the description of the LOGMODE operand of the NIB macro instruction for more information.

Examples

```
INQ1      INQUIRE   RPL=RPL1,OPTCD=APPSTAT,NIB=NIB1
TST1      TESTCB    RPL=RPL1,FDBK=0
          BE        ACTIVE
          .
          .
          .
NIB1      NIB       NAME=PGM1
```

INQ1 determines whether PGM1 is active and accepting logons. The answer is returned in RPL1's FDBK field. TST1 and the branch instruction cause a branch to ACTIVE if the application program is active and accepting logons.

```
INQ2        INQUIRE     RPL=RPL2,OPTCD=LOGONMSG,
                        ACB=ACB1,NIB=NIB2,
                        AREA=LGNMSG,AREALEN=100
   .
   .
   .
NIB2        NIB         NAME=LU2
LGNMSG      DS          CL100
```

INQ2 obtains the data portion of the logon that was sent from the logical unit whose symbolic name is contained in NIB2 and that was directed to the application program represented by ACB1. This data is placed in the area designated as LGNMSG.

Return of Status Information

When the INQUIRE operation is completed, the following RPL fields are set:

If INQUIRE (OPTCD=APPSTAT) has been completed normally, as indicated in register 15, the FDBK field is set as shown above.

If INQUIRE (all versions except OPTCD=APPSTAT) has been completed normally, the RECLEN field indicates the number of bytes of data that have been placed in the work area designated by the AREA field. If INQUIRE was completed successfully but the FDBK2 field indicates that the work area was too small (FDBK2=5), RECLEN indicates the required length.

The value 26 (decimal) is set in the REQ field indicating an INQUIRE request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

INTRPRET–Interpret an Input Sequence

For each terminal, INTRPRET allows application programs to use interpret tables that are specified by the user and maintained by VTAM, rather than tables that are created and maintained by each application program.

During VTAM definition, the user identifies each terminal (logical unit, BSC or start-stop terminal, or local non-SNA 3270) in its teleprocessing network, and optionally associates an *interpret table* with each one. The interpret table contains one or more variable-length sequences that the terminal is capable of sending–such as graphic characters, tab characters, or program function key characters. With each of these sequences, the user specifies a corresponding 8-byte sequence (or the address of an user-written routine that generates an 8-byte sequence). An application program issuing INTRPRET identifies the terminal and provides a particular sequence received from the terminal; VTAM, if it finds that sequence in the interpret table for that terminal, returns the corresponding sequence to the application program.

As an example, assume that the user defines the following interpret tables for two terminals, T2741 and T3270:

T2741's interpret table		*T3270's interpret table*	
_Logon.	LOGON	LGN	LOGON
Repeat last xmission.	REPEATLT	#	REPEATLT
Stop.	STOP	@	LIST

If an application program receives the sequence "Repeat last xmission." from T2741, INTRPRET (if provided with the sequence and the identity of the terminal) would return the sequence "REPEATLT" to the application program. If the application program identifies T3270 and provides the sequence "#" to INTRPRET, INTRPRET would return the corresponding sequence–in this case, another "REPEATLT"–to the application program.

Name	*Operation*	*Operands*
[symbol]	**INTRPRET**	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address

Indicates the location of the RPL from which INTRPRET obtains needed information from the application program, and into which it returns completion status information.

rpl field name=new value

Indicates an RPL field to be modified and the new value that is to be contained or represented within it. To avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the INTRPRET macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction or it can indicate a register. ARG=(*register*) can also be coded.

The following RPL operands apply to an INTRPRET macro instruction:

ACB=acb address
Indicates the ACB that identifies the application program.

ARG=(register)
Indicates the register containing the CID of the terminal. VTAM looks for an interpret table for this terminal.

NIB=nib address
Indicates the NIB whose NAME field identifies the terminal. VTAM looks for an interpret table for this terminal. If the NIB field does not indicate a NIB address, the ARG field must contain a CID.

AREA=data address
Indicates the data area containing the sequence being submitted to VTAM for interpretation.

RECLEN=data length
Indicates how many bytes are being submitted to VTAM for interpretation.

AAREA=data area address
Indicates the data area where VTAM is to place the interpreted sequence.

AAREALN=data area length
Indicates the capacity of the data area where VTAM is to place the interpreted sequence. This value should be at least 8.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) INTRPRET macro instruction is completed. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When the SYN option code is set, control is returned to the application program when the INTRPRET macro instruction has been completed. The macro instruction is completed as soon as the data has been placed in the application program's storage area. When ASY is set, control is returned as soon as VTAM has accepted the request. Once the INTRPRET macro instruction has been completed, the ECB is posted or the RPL exit routine is scheduled, depending on the setting of the ECB-EXIT field.

Example

```
INT1      INTRPRET  RPL=RPL1,
                    NIB=NIB6,AREA=INSEQ,RECLEN=(3),
                    AAREA=OUTSEQ,AAREALN=8
          .
          .
          .
RPL1      RPL
INSEQ     DS        CL180
NIB6      NIB       NAME=TERM1
OUTSEQ    DS        CL8
```

An application program has read a block of data from TERM1 and issues INT1 to interpret that data. NIB6 identifies the terminal, (and therefore, the interpret table to be used), AREA indicates the data area containing the data to be interpreted (INSEQ), and RECLEN indicates the amount of data to be interpreted. Note that if INTRPRET uses the same RPL that was used to read the data, the NIB-ARG field, the AREA field, and the RECLEN field are already correctly set.

Upon completion of INT1, the corresponding sequence is placed in the data area identified by the AAREA field (OUTSEQ). Although two separate data areas have been provided in this example for the "input" data (INSEQ) and the "output" data (OUTSEQ), there is no reason why the same data area could not be used.

Return of Status Information

When the INTRPRET operation is completed, these RPL fields are set:

If the FDBK2 field indicates that INTRPRET failed because the data to be placed in the AAREA work area would not fit (FDBK2=5), the ARECLEN field contains the number of bytes required to hold the data. If INTRPRET was completed successfully, the ARECLEN field indicates how many bytes of data have actually been placed in the AAREA work area.

The value 27 (decimal) is set in the REQ field, indicating an INTRPRET request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

LDO—Create a Logical Device Order (Basic Mode Only)

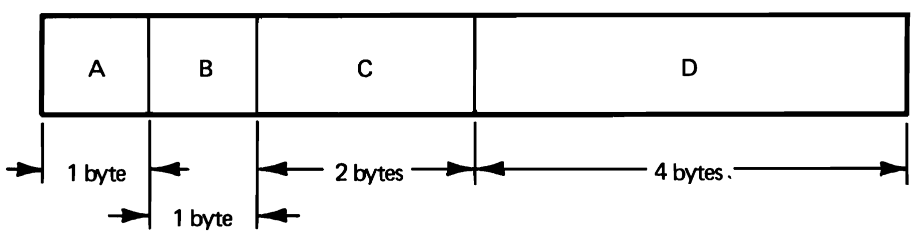
With the READ, WRITE, SOLICIT, and RESET macro instructions, the application program can perform all but a few of the I/O operations provided by VTAM. To request any of the following I/O operations, however, the application program must use the DO and LDO macro instructions:

- Copy the contents of a remotely attached 3277 Display Station buffer to the buffer of any printer or display unit attached via the same control unit. Use the COPYLBM or COPYLBT LDOs.
- Read the entire contents of a 3270 display unit's buffer. (To simply read the data that the terminal operator sends, use the READ macro instruction.) Use the READBUF LDO.
- Send a positive or negative acknowledgment accompanied by leading graphic characters, to a System/3 or System/370 CPU, and then read a block of data from it. Use the WRTPRLG or WRTNRLG LDOs.
- Write data beginning with a block of heading characters to a System/3 or System/370 CPU. Use the WRTHDR LDO.
- End an NCP session with a terminal. Use the DISCONCT LDO.
- Erase the entire display screen of a 3270 display station (or a 2265 display station attached to a 2770 Data Communication System) and write a block of data, or erase only the unprotected portion of a 3270 display station screen and write no data. Although these operations are available through the WRITE macro instruction, the latter does not allow erasure to be combined with a conversational WRITE operation. If the ERASELBM or ERASELBT LDOs are followed by a chained READ LDO, however, a combined erase-write-read operation can be achieved. If the EAU LDO is followed by a chained READ LDO, a combined erase-read operation can be achieved.

The LDO macro instruction generates a control block during program assembly that indicates one of the above I/O operations. The actual operation is performed when a DO macro instruction is executed.

Some LDOs can be combined to form a series of operations, much like channel command words can be combined to form a channel program.

An LDO has four parts:



A   A *command* indicator. This indicates the specific I/O operation to be performed.

B   A *chaining* indicator. A flag can be set in some of the LDOs that cause DO processing to also use the next contiguous LDO in storage.

C   A *length* indicator. This indicates the length of the data or data area. (The RPL also has corresponding data address and length fields, but these indicate the LDO address, not the data address, when the RPL is used by DO.)

D   A *data* address or a *data area* address. Depending on the command, this address indicates an area containing data, or a storage area where data is to be placed.

Although the operands corresponding to these parts are optional when the LDO macro instruction is coded, the command, and usually the data address and length indicator must be set before the DO macro instruction is executed. The LDO command descriptions below indicates whether these fields must be set. Assembler language must be used if you want to set LDO fields during program execution. You cannot use the manipulative macro instructions to modify LDO fields.

Name	*Operation*	*Operands*
[symbol]	**LDO**	**[CMD=command]** **[, ADDR=data address or data area address]** **[, LEN=data length or data area length]** **[, FLAGS=C|D]**

CMD=command

Format: After the CMD keyword, code any of the following values:

COPYLBM	WRITE	WRTNRLG	EAU
COPYLBT	WRITELBM	WRTPRLG	DISCONCT
READ	WRITELBT	ERASELBM	
READBUF	WRTHDR	ERASELBT	

Function: Indicates the specific I/O operation to be performed.

COPYLBM
This LDO causes the entire contents of a 3277 Display Station buffer to be copied to a printer or another display unit in the same remotely attached information display system. VTAM sends the copy request as a message by adding an ETX line control character at the end. This LDO applies only to remotely attached 3270 terminals.

The ADDR and LEN operands of this LDO must indicate the location and length of a data area containing (1) a 3270 copy control character and (2) the rightmost 2 bytes of the "from" device's CID. For an explanation of the copy control character, refer to *IBM 3270 Information Display System Component Description,* GA27-2749.

The ARG field of the DO macro instruction's RPL must contain the CID of the "to" device.

COPYLBT
The COPYLBT LDO performs like the COPYLBM LDO, except that after the data has been copied, VTAM waits for the receiving device's acknowledgment, and sends an EOT character after the acknowledgment is received. (The LBM and LBT in the COPYLBM and COPYLBT LDOs stand respectively for "last block in message" and "last block in transmission.")

READ
The READ LDO obtains a block of data from a System/3 or System/370 CPU and places it in a storage area in the application program.

The READ LDO causes VTAM to perform the same action that a READ macro instruction does. However, a READ LDO can be command-chained after a

WRTPRLG or WRTNRLG LDO. This allows the application program to either (1) send a negative acknowledgment to the device and then reread the data sent by it or (2) send a positive acknowledgment to the device and then read the next block of data (or EOT character) sent by it. By generating its own responses in this manner, the application program can send leading graphic characters along with the response.

If, at the time DO is executed, no solicited data is in VTAM buffers from the terminal, VTAM first solicits data from the terminal. This "implicit" solicitation operates as if a SOLICIT macro instruction had been issued.

The ARG field of the DO macro instruction's RPL must contain the CID of the device. The ADDR and LEN fields of the READ LDO must indicate the location and length of the storage area where the data is to be placed.

If the data to be placed there is too long to fit, and the TRUNC option code is in effect, the excess data is discarded. If the KEEP option is in effect instead of TRUNC, as much data as will fit is placed in the input area, and the length of the moved data is placed in RECLEN (so RECLEN=LEN), and the LDO's address is placed in the RPL's AAREA field. The excess data can be obtained with another READ LDO or with a READ macro instruction.

READBUF
The READBUF (read buffer) LDO causes the entire contents of a 3275 or 3277 Display Station's buffer to be placed in an area in the application program. VTAM sends the device-control characters required to distinguish this kind of input operation from a normal read operation (which obtains data only when the terminal operator enters data and presses ENTER). This LDO applies to both locally and remotely attached 3270 terminals.

The ARG field of the DO macro instruction's RPL must contain the CID of the sending device.

The ADDR and LEN operands of this LDO indicate the address and length of the storage area where the data is to be placed. The action taken when the data is too long to fit is the same as described above for READ.

WRITE
The WRITE LDO writes a block of data to a System/3 or System/370 CPU. For these devices, the WRITE LDO works exactly like a WRITE macro instruction with a BLK option code; an STX character is added to the beginning of the data, and an ETB line control character is added to the end. However, if a WRITE LDO is command-chained after WRTHDR LDO, (by specifying FLAGS=C on the WRTHDR LDO), this sequence is written:

S		S		E
O	heading	T	text	T
H		X		B

The ADDR and LEN operands of the WRITE LDO must indicate the location and length of the text data to be written. The ARG field of the DO macro instruction's RPL must contain the CID of the receiving device.

WRITELBM
The WRITELBM LDO writes a block of data to a System/3 or System/370 CPU. For these devices, WRITELBM works exactly like a WRITE macro instruction with

an LBM option code; an STX character is added to the beginning of the data, and an ETX character is added to the end. However, if a WRITELBM LDO is chained after a WRTHDR LDO, this sequence is written:

```
S            S          E
O  heading   T   text   T
H            X          X
```

The ADDR and LEN operands of the WRITELBM LDO must indicate the location and length of the text to be written. The ARG field of the DO macro instruction's RPL must contain the CID of the receiving device.

WRITELBT

The WRITELBT writes a block of data to a System/3 or System/370 CPU. For these devices, WRITELBT works exactly like a WRITE macro instruction with an LBT option code; the data is preceded with an STX character and followed with an ETX character, and when an acknowledgment is received from the device, an EOT character is sent. However, if a WRITELBT LDO is chained after a WRTHDR LDO, this sequence is written:

```
S            S          E   E
O  heading   T   text   T   O
H            X          X   T

          acknowledgment
          received
```

The ADDR and LEN operands of the WRITELBT LDO must indicate the location and length of the text to be written. The ARG field of the DO macro instruction's RPL must contain the CID of the receiving device.

WRTHDR

The WRTHDR LDO writes a block of heading characters to a System/3 or System/370 CPU. The heading characters are provided by the user; VTAM inserts an SOH character at the beginning of the block and an ETB character at the end.

If a WRITE, WRITELBM, or WRITELBT LDO is chained to a WRTHDR LDO (by specifying FLAGS=C on the WRTHDR LDO), the ETB character is not inserted after the heading. See the above descriptions of the WRITE, WRITELBM, and WRITELBT commands.

The ADDR and LEN operands of this LDO must indicate the location and length of the heading characters to be written. The ARG field of the RPL being used by the DO macro instruction must contain the CID of the receiving device.

WRTNRLG

The WRTNRLG LDO (write negative response with leading graphics) sends a NAK character, accompanied by up to seven leading graphic characters, to a System/3 or System/370 CPU. WRTNRLG can be used only if it is command-chained before a READ LDO (by specifying FLAGS=C on the WRTNRLG LDO) and if BLOCK has been specified for the device's NIB.

The ADDR and LEN operands of the WRTNRLG LDO must indicate the location and number of graphic characters to be used. The ARG field of the DO macro instruction's RPL must contain the CID of the receiving device.

WRTPRLG
The WRTPRLG LDO (write positive response with leading graphics) sends an ACK0 or ACK1 sequence, accompanied by up to seven leading graphic characters, to a System/3 or System/370 CPU. WRTPRLG can be used only if it is command-chained before a READ LDO (by specifying FLAGS=C on the WRTPRLG LDO) and BLOCK has been specified for the device's NIB.

The ADDR and LEN operands of the WRTPRLG LDO must indicate the location and number of graphic characters to be used. The ARG field of the DO macro instruction's RPL must contain the CID of the receiving device.

ERASELBM
The ERASELBM LDO (erase, write last block of message) erases the screen of a 3270 display station or the screen of a 2265 display station attached to a 2770 Data Communication System. It then writes a block of data ending with STX to the terminal. A READ LDO can be chained after an ERASELBM LDO.

The ADDR and LEN operands of the ERASELBM LDO must indicate the location and length of the data to be written. The ARG field of the DO macro instruction's RPL must contain the CID of the terminal.

ERASELBT
The ERASEBLT LDO (erase, write last block of transmission) works exactly like the ERASELBM LDO, except that after the data is sent to the terminal and an acknowledgement is received, an EOT character is sent. A READ LDO can be chained after an ERASELBT LDO.

EAU
The EAU LDO (erase all unprotected) erases the unprotected portion of a 3270 display station's screen. No data is written to the terminal. A READ LDO can be chained after an EAU LDO.

The ARG field of the DO macro instruction's RPL must contain the CID of the terminal.

DISCONCT
The DISCONCT LDO (disconnect) sends an EOT to the terminal and terminates the NCP session with the terminal. The ARG field of the DO macro instruction's RPL must contain the CID of the terminal.

ADDR=data address or data area address
Indicates the location of the data or data area to be used when the LDO is processed.

For COPYLBM and COPYLBT LDOs, ADDR points to a 3270 copy control character and rightmost 2 bytes of the "from" device's CID. For the READ and READBUF LDOs, ADDR indicates where the data obtained by these LDOs is to be placed. For the output LDOs, ADDR indicates the location of the data that is to be written to a device.

If you omit this operand, the ADDR field is set to 0. Register notation is not permitted.

LEN=data length or data area length
Indicates the length (in bytes) of the data or data area specified in ADDR.

For COPYLBM and COPYLBT LDOs, this value should always be set to 3. For READ and READBUF LDOs, VTAM uses this value to determine whether the data to be placed there is too big to fit. For all output LDOs, LEN indicates how many bytes of data are to be written.

Format: The maximum length you can specify is 32,767 bytes. If you omit this operand, the LEN field is set to 0.

Register notation is not permitted.

FLAGS=C | D

Indicates the action that the DO macro instruction is to take after it has used this LDO. The presence of this operand indicates that DO is to continue with the next contiguous LDO in storage. FLAGS=C (command chaining) indicates that the entire LDO is to be used. FLAGS=D (data chaining) indicates that only the ADDR and LEN fields of the next LDO are to be used. The absence of this operand indicates to DO that no further LDOs are to be used. A maximum of 100 LDO's may be chained together.

Note: *A WRTNRLG or a WRTPRLG LDO must be command-chained to a READ LDO; a READ LDO cannot be command-chained to another READ LDO, and a COPYLBM or a COPYLBT LDO cannot be command-chained to any other LDO.*

Examples

The following example illustrates the use of the COPYLBM LDO.

Assume that the CID of the 'to' device is already in the ARG field of the DO macro's RPL, and that the CID of the 'from' device (the CID that must be manipulated) is in the CID field of NIB1.

```
PRIME       SHOWCB   NIB=NIB1,AREA=TEMP,LENGTH=4,FIELDS=CID
            MVC      CPYSCRN1+1(2),TEMP+2
CPYSCRN     DO       RPL=RPL1,AREA=LDO1
             .
             .
             .
LDO1        LDO      CMD=COPYLBM,ADDR=CPYSCRN1,LEN=3
TEMP        DS       F        (TEMP=WORK AREA FOR 'FROM' CID)
CPYSCRN1    DC       X'630000' (63=A COPY CONTROL CHARACTER,
*                    0000=FINAL AREA FOR RIGHT
*                    HALF OF 'FROM' CID)
```

The purpose of the two instructions at PRIME is to obtain the CID of a 'from' device (from NIB1 into TEMP) and place the rightmost 2 bytes of the CID into a data area pointed to by LDO1. When CPYSCRN is executed, the device whose CID is in RPL1's ARG field will be the recipient of the copy operation.

The next example shows how a READBUF LDO might be used.

```
READ2       DO       RPL=RPL2,ARG=(7),AREA=READLDO
             .
             .
             .
READLDO     LDO      CMD=READBUF,ADDR=WORKAREA,LEN=480
WORKAREA    DS       CL480
```

When READ2 is executed, register 7 must contain the CID of a 3270 display unit. VTAM will obtain the entire contents of that device's buffer and place it in WORKAREA.

The following example illustrates the use of the WRTHDR LDO.

```
WRITEIT    DO    RPL=RPL3,ARG=(8),AREA=HDRLDO
                 .
                 .
                 .
HDRLDO     LDO   CMD=WRTHDR,ADDR=HDRBLOK,LEN=5,FLAGS=C
TXTLDO     LDO   CMD=WRITE,ADDR=TXTBLOK,LEN=16
HDRBLOK    DS    CL5
TXTBLOK    DS    CL200
```

When WRITEIT is executed, VTAM sends a heading block from AHDRBLOK combined with a text block from ATXTBLOK. The line control characters added by VTAM make the sequence look like this:

```
S    data     S    data     E
O    from     T    from     T
H  AHDRBLOK   X  ATXTBLOK   B
```

The following example shows how a WRTPRLG LDO can be command-chained to a READ LDO.

```
POSRSP     DO    RPL=RPL4,ARG=(9),AREA=RSPLDO
                 .
                 .
                 .
RSPLDO     LDO   CMD=WRTPRLG,ADDR=GRAPHICS,LEN=7,FLAGS=C
THENREAD   LDO   CMD=READ,ADDR=INAREA,LEN=480
GRAPHICS   DC    C'CPU3003'
```

When POSRSP is executed, register 9 must contain the CID of a device. VTAM sends a positive response (ACK0 or ACK1) to the device, accompanied by seven leading graphic characters from GRAPHICS. The next LDO causes VTAM to read the next block of data from the device.

MODCB–Modify the Contents of Control Block Fields

MODCB modifies the contents of one or more fields in an ACB, EXLST, RPL, or NIB control block. MODCB works with control blocks created either with declarative macro instructions or with the GENCB macro instruction.

The user of the MODCB macro instruction indicates the location of an ACB, EXLST, RPL, or NIB, the fields within that control block to be modified, and the new values that are to be placed or represented in these fields.

Any field whose contents can be set with the ACB, EXLST, RPL, or NIB macro instruction can be modified by the MODCB macro instruction. The operands used to do this are the same as those used when the control block is created.

The following restrictions apply to the use of MODCB:

- An ACB cannot be modified after an OPEN macro has been issued for it.
- An exit list (EXLST) cannot have exits added to it with the MODCB macro instruction. If an exit list field is not specified in the EXLST macro instruction, do not attempt to modify that field with a MODCB macro instruction. MODCB can, however, be used to change dummy exit addresses to valid addresses.
- An RPL cannot be modified while a request using that RPL is pending, that is, while the RPL is active.
- A NIB should not be modified while its address is in the NIB field of an active RPL.
- The AM field of the ACB, EXLST, and RPL control blocks cannot be modified. Once a control block has been generated in a VTAM-compatible form, it cannot later be modified for use with another access method.

List, generate, and execute forms of the MODCB macro instruction are available; they are designated by the MF operand.

Because there are a large variety of formats in which the various MODCB operand values can be specified, the operand format specifications have been tabulated in Appendix E, and do not appear here.

Note: *For terminals that use basic-mode, MODCB NIB PROC and USERFLD data can be changed with MODCB; however, VTAM must also be notified of this change with a CHANGE macro instruction. See the description of the CHANGE macro instruction for more information.*

Name	*Operation*	*Operands*
[symbol]	**MODCB**	**AM=VTAM** **{ , ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address }** **, field name=new value ...** **[, MF=list, generate, or execute form parameters]**

AM=VTAM

Identifies this macro instruction as a VTAM macro instruction. This operand is required.

ACB=acb address
EXLST=exit list address
RPL=rpl address
NIB=nib address

Indicates the type and location of the control block whose fields are to be modified.

field name=new value

Indicates a field in the control block to be modified and the new value that is to be contained or represented within it.

For *field name,* code the keyword of any operand that can be coded in the macro instruction corresponding to the ACB, EXLST, RPL, or NIB operand. If RPL=RPL1 is coded, for example, the keyword of any operand in the RPL macro instruction can be coded. ARG=(register) can also be coded.

For *new value,* EXLST, RPL, or NIB macro instruction, or use one of the formats indicated in Appendix E.

MF=list, generate, or execute form parameters

Indicates that a list, generate, or execute form of MODCB is to be used. Omitting this operand causes the standard form of MODCB to be used. See Appendix F for a description of the nonstandard forms of MODCB.

Example

```
MOD1        MODCB       RPL=(5),OPTCD=(ASY,SPEC,CS),AREA=(6),
                        AM=VTAM
```

MOD1 activates the ASY, SPEC, and CS option codes in an RPL. The settings for the other option codes are not affected. The address of this RPL must be in register 5 when MOD1 is executed.

Return of Status Information

After MODCB processing is completed, register 15 indicates whether or not the operation completed successfully. If the operation completed successfully, register 15 is set to 0; if it completed unsuccessfully, register 15 is set to either 4, 8, or 12 (DOS/VS only). If it is set to 4 or 12, register 0 is also set indicating the specific nature of the error (see Appendix D).

NIB—Create a Node Initialization Block

The NIB generated by the NIB macro instruction is used by the program to identify which terminal (logical unit, BSC or start-stop terminal, or local non-SNA 3270) is to be connected when an OPNDST macro instruction is executed. It also indicates how VTAM is to handle subsequent communication between the program and that terminal. In this sense a NIB works like an RPL, in that both contain information that governs I/O requests. But the information in a NIB relates to the *terminal* the NIB represents and governs all communication directed at that *terminal.* (The RPL, in contrast, supplies additional information relating to the request itself, such as the location of data to be written to a terminal or whether or not the request is to be handled asynchronously.)

When OPNDST is issued, the NIB field of its RPL points to a NIB. Once connection is established, VTAM associates the terminal with information contained in the NIB—information that is placed in the NIB by the USERFLD, MODE, and PROC operands of the NIB macro instruction. This association continues as long as the terminal remains connected, even though the storage containing the NIB can be used for other purposes as soon as OPNDST is completed; therefore, the same NIB can be used by other macro instructions, (for example, INQUIRE). If the USERFLD or PROC information is to be altered during that time (basic-mode only), the MODCB macro instruction or a new NIB must be used to make the appropriate changes in the USERFLD or PROC fields of a NIB (either the original or a new one), and the CHANGE macro instruction must be used to make these modifications effective.

NIBs can be grouped together into lists. When OPNDST (OPTCD=ACQUIRE) and SIMLOGON requests are directed towards a NIB that is the first in a NIB list, VTAM considers all of the terminals represented in the NIB list to be the objects of the request, not just the terminal represented by the first NIB.

A field called the CID field is part of every NIB. It is not represented in the NIB macro instruction because its contents cannot be set by the application program. When the terminal represented by the CID is connected to the program, VTAM generates an equivalent of the terminal's symbolic name and places it both in the NIB's CID field and in the ARG field of the RPL being used by the OPNDST macro instruction. For NIB lists, the CID placed in the ARG field is not meaningful. Subsequent I/O requests directed toward that specific terminal must have this CID in the I/O request's RPL.

The NIB macro instruction causes the NIB to be built during program assembly; the NIB macro instruction is not executable. The NIB is built on a fullword boundary. A NIB can be built during program execution with the GENCB macro instruction.

Name	*Operation*	*Operands*
[symbol]	NIB	[NAME=name in resource definition table] [,USERFLD=fullword of data] [,LISTEND=YES|NO] [,MODE=BASIC|RECORD] [,SDT=APPL|SYSTEM] [,EXLST=exit list address] [,RESPLIM=1| response limit] [, LOGMODE=0|C' '|logon mode] [, BNDAREA=0| bind area address] [[CA|CS|RPLC] [,NDFASYX|DFASYX] [,NRESPX|RESPX] [,NCONFTXT|CONFTXT] [,KEEP|TRUNC] [, BLOCK|MSG|TRANS|CONT] [,LGOUT|NLGOUT] [,LGIN|NLGIN] ,PROC=([,TMFLL|NTMFLL]) [,NEIB|EIB] [,TIMEOUT|NTIMEOUT] [,ERPIN|NERPIN] [,ERPOUT|NERPOUT] [,NMONITOR|MONITOR] [,NELC|ELC] [,NBINARY|BINARY]]

NAME=name in resource definition table

Associates the NIB with a resource represented in the resource definition table. (The resource definition table is built by the user during VTAM definition.)

Format: Use the name of the PU, LU, CLUSTER, TERMINAL, LINE, COMP, LOCAL, or APPL entry that represents one or more terminals or application programs in the resource definition table. Using unframed EBCDIC characters, code this name as it appears in the resource definition table.

Example: NAME=LU13

Note: *Although this operand is optional, the NAME field should be set by the time a CLSDST, OPNDST, SIMLOGON, INTRPRET, or INQUIRE macro instruction is issued for this NIB. One exception: When OPNDST with an ACCEPT processing option and an ANY option code is issued, the NAME field need not be set, since* VTAM *will place the name of the connected terminal in this field.*

If you omit this operand, the entire 8-byte NAME field is set to EBCDIC blanks.

USERFLD=full word of data

Indicates any 4 bytes of data that the application program wants to associate with the terminal represented by this NIB. When you subsequently issue I/O requests for the terminal, VTAM will place whatever data you have set in USERFLD into the USER field of the I/O request's RPL.

The 4 bytes of data can be anything the application program chooses to associate with the terminal. It can be the program's own version of the terminal's symbolic name. This would be useful in the case of a RECEIVE or READ macro with OPTCD=ANY, since the setting of the USER field in the macro's RPL provides an efficient way for the program to establish the identity of the terminals from which the data was just obtained.

Format: Code the 4 bytes of data in either character, fixed-point, or hexadecimal format, or, if an address is desired, code it as an A-type or V-type address constant. Register notation cannot be used.

Examples:

USERFLD=C'LU01'
USERFLD=F'100'
USERFLD=X'00043E0'
USERFLD=A(RTN1)
USERFLD=V(EXTRTN)

Note: *For basic mode only, use the MODCB and CHANGE macro instructions to change the contents of the USERFLD field after an OPNDST macro instruction has been issued for the NIB.*

If you omit this operand, the USERFLD field is set to zero.

LISTEND=YES|NO

Allows the application program to group NIBs into lists. LISTEND=YES indicates that this NIB is the last in a list (or is an isolated NIB not part of a list). LISTEND=NO indicates that this NIB and the NIB immediately following it in storage are part of a NIB list. Any number of NIBs can be grouped together by specifying LISTEND=NO for each one except the last.

NIB lists are used by the OPNDST macro with an ACQUIRE option code and by the SIMLOGON macro. VTAM considers the terminals represented by the entire list as objects of the OPNDST or SIMLOGON macros.

Example: The following use of the LISTEND operand effectively groups the Boston NIBs into one group, the Chicago NIBs into another, and defines the Portland NIB as a "list" of one.

```
BOSTON      NIB   NAME=BOSTON1,MODE=RECORD,LISTEND=NO
            NIB   NAME=BOSTON2,MODE=RECORD,LISTEND=YES
CHICAGO     NIB   NAME=CHICAGO1,MODE=RECORD,LISTEND=NO
            NIB   NAME=CHICAGO2,MODE=RECORD,LISTEND=NO
            NIB   NAME=CHICAGO3,MODE=RECORD,LISTEND=YES
PORTLAND    NIB   NAME=PORTLAND,MODE=RECORD,LISTEND=YES
```

MODE=BASIC|RECORD

Indicates whether basic-mode macro instructions (such as READ and WRITE) or record-mode macro instructions (such as SEND and RECEIVE) are to be used to communicate with the terminal. Except for 3270 terminals, the application program has no choice; MODE=BASIC must be specified for all BSC and start-stop terminals, and MODE=RECORD must be specified for all logical units. 3270 terminals can be handled in either mode.

The application program can be designed to handle both modes, since the INQUIRE macro instruction (OPTCD=DEVCHAR) can be used before connection to determine whether a logical unit or a specific type of BSC or start-stop terminal has logged on. A simpler and better procedure, however, would be to maintain one ACB (application program, in effect) for BSC and start-stop terminals and another ACB for logical units. Using separate ACBs eliminates the need to repeatedly test and set the mode type.

Note: *This field must be filled in before the OPNDST macro instruction is issued.*

SDT=APPL|SYSTEM (Record mode only)

Indicates whether the application program or VTAM is to send the first Start Data Traffic (SDT) command to the logical unit. If SDT=SYSTEM is used, VTAM automatically sends an SDT command as part of the connection process before posting the OPNDST RPL complete. If SDT=APPL is coded, VTAM does not send an SDT command until the application program tells it to do so (by issuing a SESSIONC macro instruction with CONTROL=SDT). The use of this operand is determined by the transmission services profile that is specified in the session parameters used for connection (see Appendix J).

The SDT command is used to permit the flow of messages and responses between the application program and the logical unit. See the SESSIONC macro instruction for more information.

EXLST=exit list address (Record mode only)

Indicates an EXLST control block that contains the address of a DFASY, RESP, or SCIP exit routine (or contains the addresses of any combination of these exit routines).

Exit routines indicated by a NIB (NIB-oriented exit routines) are scheduled when VTAM receives input from the logical unit associated with the NIB. If input is received and no NIB-oriented exit routine is specified, VTAM then satisfies any pending RECEIVE macros or schedules the appropriate ACB-oriented exit routine (if any). See the descriptions of the DFASY, RESP, and SCIP exit routines, in the EXLST macro instruction description, for more information on the scheduling of these exit routines.

Figure 5 shows two sets of NIB-oriented exit routine addresses and one set of ACB-oriented exit routine addresses. When input from the logical unit associated with NIB1 arrives, the appropriate EXLST1 exit routine is scheduled. When input from the logical unit associated with NIB2 arrives, VTAM checks EXLST2 for the appropriate exit routine. If there is no exit routine specified (which in this example would be true if the input was a response, since EXLST2 has no RESP entry), VTAM satisfies any pending RECEIVE macros or checks for an ACB-oriented exit routine address in EXLSTA. When input from any other logical unit arrives, VTAM uses EXLSTA.

Note: *If you omit this operand, the NIB's EXLST field is set to 0.*

RESPLIM=1|response limit (Record mode only)

Indicates the maximum number of responded output requests that can be pending at one time. A responded output request is a SEND with POST=RESP, STYPE=REQ, and CONTROL specifying data or a normal-flow, data flow control command. If RESPLIM=0 is coded, VTAM imposes no limit on the number of pending responded output requests.

Note: *If this operand is omitted, the RESPLIM field is set to 1. The maximum value that can be coded is 65535.*

LOGMODE=0|C' '|logmode (Record mode only)

The LOGMODE field in a NIB that is pointed to by the NIB field of an RPL is used by the INQUIRE, OPNDST, CLSDST with OPTCD=PASS, and SIMLOGON macro instructions. If a bind area is used with an OPNDST macro instruction, it will override any logon mode that is also present. Figure 6 summarizes the use of the LOGMODE and BNDAREA operands.

When used by the INQUIRE macro:

- LOGMODE=0 indicates that VTAM is to take the session parameters associated with a pending logon and any logon data, if present, and place them in the field pointed to by the AREA field of the RPL. The AREALEN field of the RPL must specify the length of the area.

  Return codes are returned if a match is not found (RTNCD=20, FDBK2=75) or if there is no pending logon (RTNCD=20, FDBK2=76).

- LOGMODE=C' ' indicates to VTAM that default session parameters (those making up the first entry in the logon mode table for the logical unit named in the NIB) are to be returned in the field pointed to by the AREA field. The AREALEN field of the RPL must specify the length of the area.

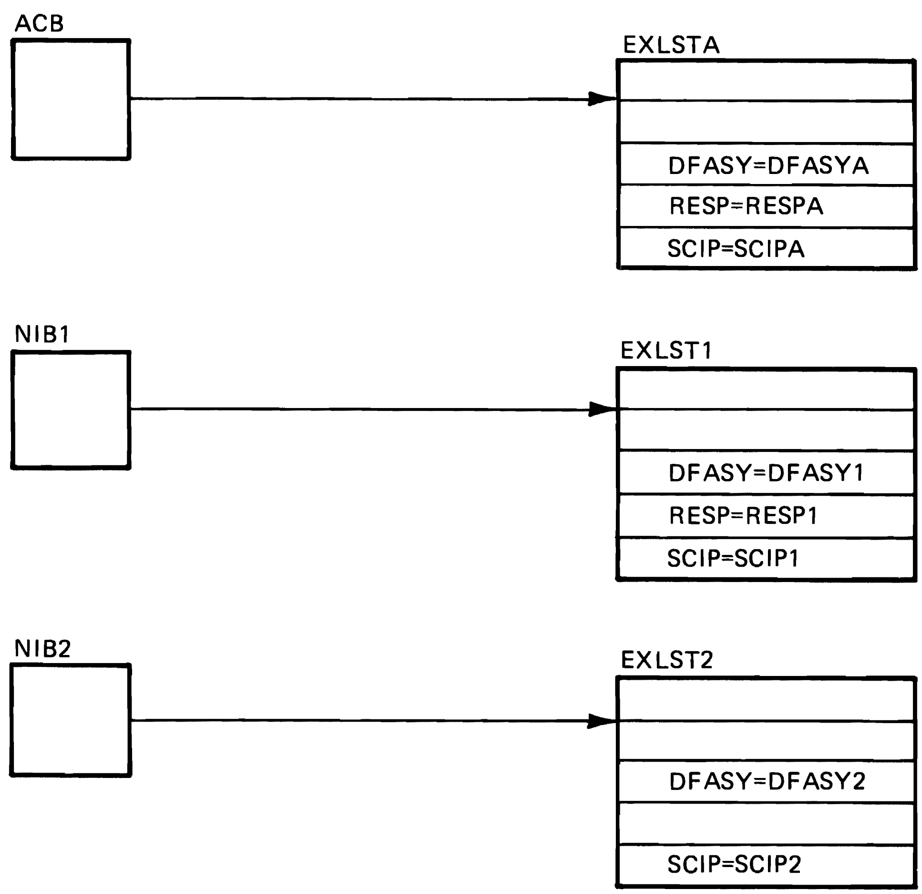


Figure 5. ACB-Oriented and NIB-Oriented Exit Routines

If the NIB[1] pointed to by an RPL, specifies . . .		then INQUIRE places in the RPLs AREA field the session parameters associated with . . .	then OPNDST with OPTCD= ACCEPT uses the session parameters associated with . . .	then OPNDST with OPTCD= ACQUIRE uses the session parameters associated with . . .	then SIMLOGON or CLSDST with OPTCD= PASS creates a pending logon that requests . . .
LOGMODE=	BNDAREA=[2]				
0	0	the first pending logon for the logical unit. (If there is no pending logon, or if the session parameters cannot be determined for this logon an error code is returned.)[4]	the first pending logon that will satisfy the OPNDST ACCEPT. (If the session parameters cannot be determined for this logon, an error code is returned.)[4]	the default logon mode in the logon mode table.	the default logon mode in the logon mode table
C' '	0	the default logon mode in the logon mode table.	the default logon mode in the logon mode table.	the default logon mode in the logon mode table.	the default logon mode in the logon mode table
logon mode	0	the logon mode specified. (If the logon mode is not in the logon mode table, an error code is returned.)	the logon mode specified. (If the logon mode is not in the logon mode table, an error code is returned.)	the logon mode specified. (If the logon mode is not in the logon mode table, an error code is returned.)	the logon mode specified
	bind area address[3]	see Note 5	the bind area specified	the bind area specified	see Note 5

[1] If the NAME field of the NIB does not specify a logical unit, the LOGMODE and BNDAREA operands should not be used or should be set to 0. An IBM-supplied set of non-SNA session parameters will be used.

[2] If the BNDAREA operand specifies any value but 0, the LOGMODE operand will be ignored.

[3] The BNDAREA operand specifies the address of an area in the application program that contains user-supplied session parameters.

[4] A pending logon can originate from:

- A VARY LOGON command entered by the network operator (automatic logon)
- A logon entered by the terminal user
- An Initiate Self command issued by the logical unit
- A CLSDST with OPTCD=PASS macro instruction
- A SIMLOGON macro instruction

Note that a VTAM application program is unable to determine the source of a pending logon. The application programmer must know in advance the source of pending logons and the logon modes that are possible and code the application program accordingly.

[5] For INQUIRE, SIMLOGON, or CLSDST with OPTCD=PASS, the *bind area address* is ignored and the session parameters returned are determined by the LOGMODE operand.

Note: See Appendix J for a list of the logon modes and associated session parameters in the IBM-supplied logon mode table.

Figure 6. The Relationship Between the LOGMODE and BNDAREA Operands of the NIB

- LOGMODE=*logon mode name* indicates to VTAM the logon mode name with which it is to search the logon mode table defined for the logical unit named in the NIB. If a match is found, the session parameters are returned in the field pointed to by the AREA field of the RPL. The AREALEN field of the RPL must specify the length of AREA. If a match is not found, return codes are returned (RTNCD=20, FDBK2=75).

Note: *In each of the above cases, if the NIB does not name a logical unit, a standard set of non-SNA session parameters is returned.*

When used by OPNDST (OPTCD=ACCEPT) macro instruction:

- LOGMODE=0 and BNDAREA=0 indicate that VTAM is to take the session parameters from a pending logon and construct a Bind command which will be sent to the logical unit establishing the connection. If there is no pending logon, return codes are also returned (RTNCD=20, FDBK2=76).
- LOGMODE=*logon mode name* and BNDAREA=0 indicate to VTAM the logon mode name with which it is to search the logon mode table defined for the logical unit named in the NIB. If a match is found, the session parameters associated with that logon mode name are used to construct a Bind command, which is sent to the logical unit establishing the connection. Return codes are returned if a match is not found (RTNCD=20, FDBK2=75) or if there is no pending logon (RTNCD=20, FDBK2=76). The BNDAREA operand must be 0.
- LOGMODE=C' ' and BNDAREA=0 indicate that VTAM is to take default session parameters (those making up the first entry in the logon mode table) and construct a Bind command which will be sent to the logical unit establishing connection. If there is no pending logon, return codes are also returned (RTNCD=20, FDBK2=76).
- LOGMODE=0 | C' ' | *logon mode name* and BNDAREA=*bind area address* indicate that the session parameters in the specified bind area are to be used to construct the Bind command. The LOGMODE operand is ignored. Return codes are returned if there is no pending logon (RTNCD=20, FDBK2=76).

When used by the OPNDST (OPTCD=ACQUIRE) macro instruction:

- LOGMODE=0|C' ' and BNDAREA=0 indicate to VTAM that default session parameters (those making up the first entry in the logon mode table defined for the logical unit named in the NIB) are to be used to construct the Bind command, which is sent to the logical unit establishing the connection.
- LOGMODE=*logon mode name* and BNDAREA=0 indicate to VTAM the logon mode name with which it is to search the logon mode table defined for the logical unit named in the NIB. If a match is found, the session parameters associated with that logon mode name are used to construct a Bind command, which is sent to the logical unit establishing the connection. The BNDAREA operand must be 0. If a match is not found, return codes (RTNCD=20, FDBK2=75) are returned.
- LOGMODE=0|C' ' | *logon mode name* and BNDAREA=*bind area address* indicate that the session parameters in the specified bind area are to be used to construct the Bind command. The LOGMODE operand is ignored.

When used by the SIMLOGON or CLSDST with OPTCD=PASS macro instruction:

- LOGMODE=0 |C' ' indicates to VTAM that default session parameters (those making up the first entry in the logon mode table for the logical unit named in the NIB) are to be used as a part of the pending logon.

- LOGMODE=*logon mode name* indicates to VTAM the logon mode name with which it is to search the logon mode table for the logical unit named in the NIB. If a match is found, the session parameters associated with that logon mode name are to be used as a part of the pending logon.

BNDAREA=0|bind area address (Record mode only)

Permits the application program to explicitly specify a set of session parameters that VTAM is to use in constructing a Bind command that will be sent to a logical unit. The BNDAREA field in a NIB that is pointed to by the NIB field of an RPL is used by the OPNDST (OPTCD=ACCEPT or ACQUIRE) macro instruction. Session parameters specified in a bind area override session parameters available from any other source. Figure 5 summarizes the use of the LOGMODE and BNDAREA operands.

When used by the OPNDST macro instruction:

- BNDAREA=0 indicates that there is no bind area defined and that VTAM should use the LOGMODE field in the NIB to determine the session parameters to be used or use those session parameters associated with a pending logon. See Figure 5.
- BNDAREA=*bind area address* indicates the location of a bind area that contains the session parameters that are to be used by VTAM to construct the Bind command, which is sent to the logical unit to establish the connection. The application program is responsible for placing the appropriate session parameters into the bind area. When the BNDAREA operand specifies an address, the LOGMODE operand is ignored.

PROC=processing option|(processing option,. . .)

Indicates options VTAM is to follow for subsequent I/O requests involving the terminal connected using this NIB.

Format: Code as indicated in the assembler format table above. The parentheses can be omitted if only one option code is selected.

```
NIB          NAME=TERM13,MODE=RECORD,
             PROC=(DFASYX,RESPX,CONFTXT)

NIB          NAME=TERM14,MODE=BASIC,
             PROC=BLOCK
```

Note: *Not all processing options are valid for all types of devices. See Figure 7 at the end of this macro instruction description to see which processing options are valid for the devices supported by VTAM.*

CA|CS|RPLC

Applies for a logical unit or terminal when input is received from it that satisfies a RECEIVE macro instruction with OPTCD=ANY. This operand is useful for differentiating terminals by the type of request they may respond to. It overrides the CS/CA option codes that may have been specified in the RECEIVE's RPL, but not the RPL of any other type of macro instruction.

CS specifies that the connection should be put into continue-specific mode after this receive is completed. It can be used when a terminal may not respond to any subsequent RECEIVE (OPTCD=ANY) macros. This might be the case if the terminal normally sends multiple lines per transaction.

CA specifies that the connection should be put into or remain in continue-any mode after the receive operation is completed. It can be used when a terminal can always respond to a RECEIVE (OPTCD=ANY) macro. This might be the case if the terminal normally sends no more than one line per transaction.

RPLC, the default, specifies that the CS/CA option code in the RECEIVE RPL should be used when switching continue modes.

NDFASYX|DFASYX (Record mode only)
Indicates whether an ACB DFASY exit routine is to be scheduled when expedited-flow messages arrive in VTAM's buffers from a logical unit.

When DFASYX is set for the logical unit's NIB and no other restrictions prevent the scheduling of the ACB DFASY exit routine, the exit routine is scheduled. (Even if the exit routine cannot be scheduled, a RECEIVE OPTCD=ANY, RTYPE=DFASY will never be satisfied if the connection was established with PROC=DFASYX.) If NDFASYX is specified, the exit routine is not scheduled. See the DFASY operand of the EXLST macro instruction for information about the ACB DFASY exit routine and the conditions that must exist before it can be scheduled.

NRESPX|RESPX (Record mode only)
Indicates whether an ACB RESP exit routine may be scheduled when responses arrive in VTAM's buffers from a logical unit. When RESPX is set for the logical unit's NIB and no other restrictions prevent the scheduling of the ACB RESP exit routine, the exit routine is scheduled. (Even if the exit routine cannot be scheduled, a RECEIVE OPTCD=ANY, OPTCD=ANY,RTYPE=RESP will never be satisfied if the connection was completed with PROC=RESPX.) If NRESPX is set, the ACB RESP exit routine is not scheduled. See the RESP operand of the EXLST macro instruction for information about the ACB RESP exit routine and the conditions that must exist before it can be scheduled.

NCONFTXT|CONFTXT
Indicates whether or not the data sent to or received from this terminal is to be considered as "confidential." If CONFTXT is specified, the buffers used to hold the data are cleared before they are returned to their buffer pools. For NCONFTXT, no such clearing is done.

KEEP|TRUNC
Indicates whether overlength input data is to be kept or discarded.

When TRUNC is used, VTAM fills the input data area and discards the remainder. No error condition is raised. When KEEP is used, VTAM fills the input data area and saves the remainder for subsequent RECEIVE or READ macro instructions.

In the basic mode, the RPL's FDBK field indicates the presence of excess data. If the EOB flag is set on (DATAFLG=EOB for a TESTCB macro instruction), the entire block is in the input data area and no excess data remains. If the EOB flag is set off, there is excess data.

In the record mode, the presence of excess data can be determined by comparing the RPL's AREALEN field (input area size) with the RECLEN field (amount of incoming data). If the value in RECLEN exceeds the value in AREALEN, excess data has been kept (and will be used to satisfy the next appropriate RECEIVE).

In the record mode, the NIB's TRUNC-KEEP processing option is overridden if the KEEP TRUNC-NIBTK option code of the RECEIVE's RPL is set to KEEP or TRUNC. That is, the NIB's TRUNC-KEEP processing option is effective only if the NIBTK option code is set in the RPL.

BLOCK|MSG|TRANS|CONT (Basic mode only)
These control how many blocks of data are to be obtained from a BSC or start-stop terminal for a solicit operation and how acknowledgments are to be handled as each block arrives.

Solicit operations are all operations conducted by VTAM to transfer data from a terminal to VTAM buffers. Solicitation does not involve the transfer of data from VTAM buffers to the application program.

VTAM solicits data from a terminal when (1) the application program issues a SOLICIT macro instruction or (2) the application program issues a READ macro instruction with the SPEC option code in effect for the RPL. Solicitation is not performed in the latter case, however, if VTAM already holds data obtained from the terminal.

Before reading the descriptions of BLOCK, MSG, TRANS, and CONT that follow, examine Figure 7. This figure illustrates a typical data transmission from a terminal and shows how much of it is obtained each time a SOLICIT (or READ, as qualified above) is executed.

BLOCK
Either a line control response is sent to acknowledge receipt of the previous solicit operation, or polling is started. One block of data ending in an EOB line-control character (for start-stop devices) or an ETB or ETX line-control character (for BSC devices) is obtained. No response is sent when data is obtained as a result of the *current* solicit request. The data obtained by the current solicit request is acknowledged only when the next solicit request is issued.

If the terminal represented by this NIB is a BSC device, an authorization test is made when an OPNDST macro instruction is issued for this NIB. If the installation did not authorize the use of BLOCK by the application program (by so indicating in the application program's APPL entry during VTAM definition), the OPNDST macro instruction will not be executed successfully. (The use of BLOCK is restricted this way because it can result in line throughput that is very low compared to MSG, TRANS, and CONT. The low throughput results because the CPU, rather than just the communications controller, must be interrupted for each block.)

MSG
Blocks of data are continuously obtained until an EOT character (for start-stop devices) or a block containing an ETX character (for BSC devices) is received. In effect, this means that data is solicited from the terminal until an entire message has been received. For BSC devices, line-control responses are sent as each block is received, except for the last block. Its receipt is not acknowledged until the next solicit request is issued.

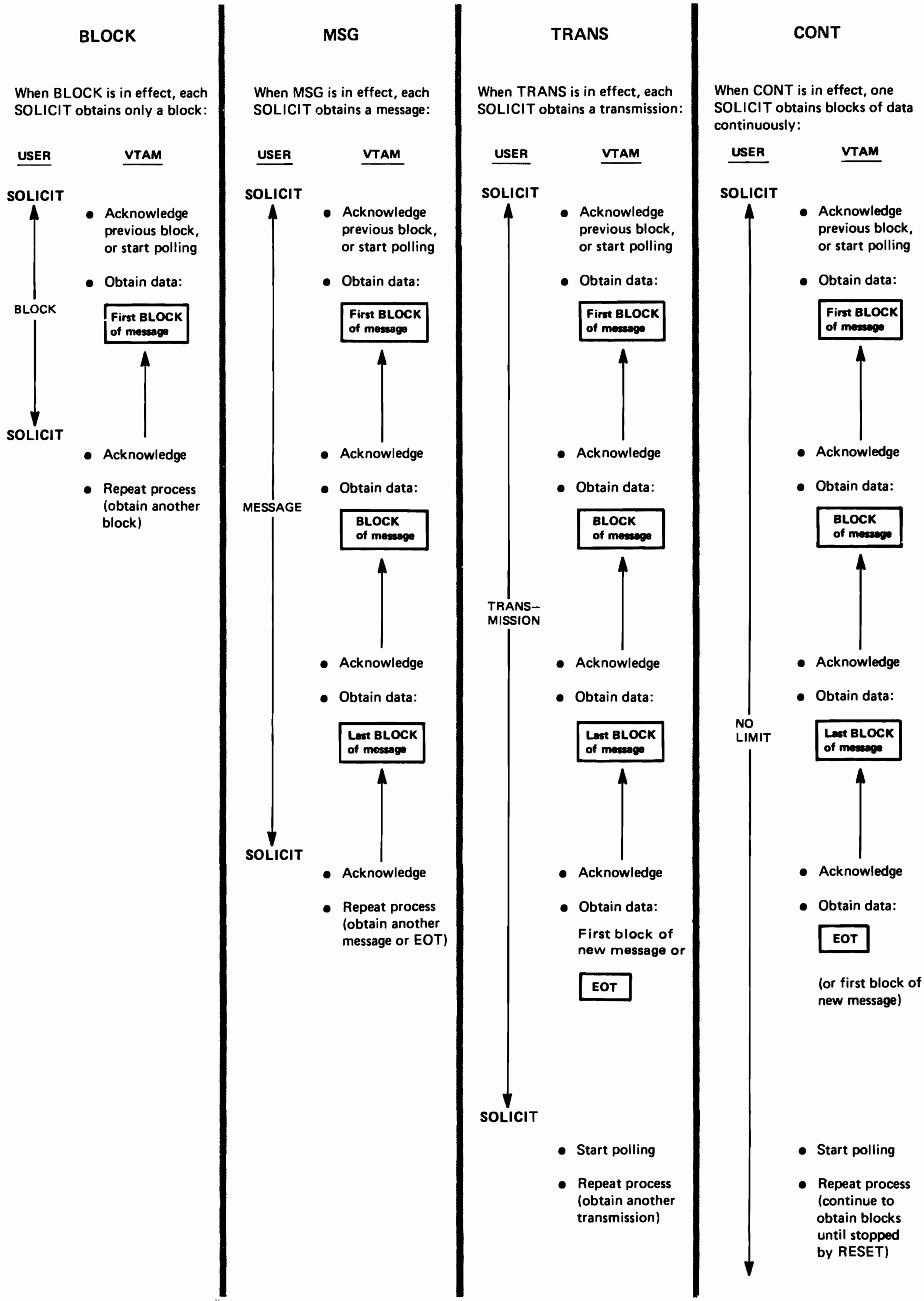


Figure 7. The Effect of BLOCK, MSG, TRANS, and CONT on Solicitation

For start-stop devices, line-control responses are sent for each block, including the last. The procedure used to solicit data from start-stop devices when the MSG processing option is specified is identical to that used when the TRANS processing option is specified.

TRANS
Blocks of data are continuously obtained until an EOT character is received. In effect, this means that data is solicited from the terminal until an entire transmission has been received. Line-control responses are sent as each block is received, including the last block. Polling is not resumed until the next solicit request is issued.

CONT
Blocks of data are continuously solicited from the terminal. Line-control responses are sent for each block received. This solicitation continues indefinitely, unless the solicit operation is canceled with the RESET macro instruction or the terminal is disconnected from the program.

LGOUT | NLGOUT (Basic mode only)
Indicates whether or not an output operation with this terminal should be considered to be in error if the terminal acknowledges receipt of the data with a response that is preceded by leading graphic characters.

When LGOUT is specified, a code is posted in the FDBK field of the WRITE request's RPL, and the leading graphic characters are held by VTAM. A READ request directed at the terminal causes the characters to be moved into the application program's storage (in the data area indicated by the AREA field of READ's RPL). If leading graphic characters are received during a conversational write operation, the characters are passed to the application program as the input data.

When NLGOUT is specified, the output operation is completed in error if leading graphic characters are received in return.

LGIN | NLGIN (Basic mode only)
Indicates whether or not an input operation with this terminal should be considered to be in error if leading graphic characters are received.

If LGIN is specified, the presence of leading graphic characters does not constitute an error condition; the application program is notified of their presence by means of a code set in the FDBK field of the input request's RPL.

When NLGIN is specified, the input operation is completed in error if leading graphic characters are detected.

TMFLL | NTMFLL (Basic mode only)
Indicates whether or not the communications controller is to insert idle device control characters into the data sent to this terminal. TMFLL allows the communications controller to insert these characters. NTMFLL suppresses the insertion of these characters, implying that the application program will be supplying its own time-fill characters. Time fill is only performed for start-stop devices, which require special timing considerations following a carriage return and horizontal tab characters. See the INHIBIT=TIMEFILL operand in the *NCP Generation and Utilities Guide.*

<u>NEIB</u> | **EIB** (Basic mode only)
Indicates whether or not the system is to insert an EIB error information byte (EIB) after every intermediate transmission block (ITB) received from this terminal. EIB indicates that an EIB is to be inserted with each intermediate transfer block; NEIB suppresses the insertion of EIBs.

If you specify insertion of EIBs, you should scan the input data for ITB characters, and analyze the next byte (which will be the EIB) to determine whether an error occurred in the intermediate block. Insertion of EIBs is appropriate when you expect that many ITBs will be required for a data block. If a transmission error occurs and you are not using EIB, you cannot determine in which ITB the error occurred, and so would have to request a retransmission of the entire block.

Note: *The presence of ITB characters in the input data does not depend on the EIB-NEIB processng option; this option only governs the presence of the EIB. The presence of the ITB character is a function of the terminal itself. See the XITB operand in the* NCP Generation and Utilities Guide.

<u>TIMEOUT</u> | **NTIMEOUT** (Basic mode only)
Indicates whether or not the communications controller should suppress any text timeout limitation that might otherwise be used with this terminal. TIMEOUT permits normal timeouts to occur; NTIMEOUT suppresses them.

When TIMEOUT is in effect, the communications controller imposes a text timeout limitation as indicated by the installation in the terminal's TERMINAL entry. (A timeout limitation means that if the interval between two successive characters sent by a terminal exceeds a set limit, the I/O operation is terminated with an error condition.) NTIMEOUT provides the application program with a means of overriding this limitation and allowing the terminal an indefinite time period between characters. See the INHIBIT=TEXTTO operand in the *NCP Generation and Utilities Guide.*

<u>ERPIN</u> | **NERPIN** (Basic mode only)
Indicates whether or not system error recovery (retry) procedures are to be used if an I/O error occurs during an *input* operation with this terminal. ERPIN means that the error recovery procedures are to be used; NERPIN means that they are not. See the INHIBIT=ERPR operand in the *NCP Generation and Utilities Guide.*

<u>ERPOUT</u> | **NERPOUT** (Basic mode only)
Indicates whether or not system error recovery (retry) procedures are to be used if an I/O error occurs during an *output* operation with this terminal. ERPOUT means that the error recovery procedures are to be used; NERPOUT means that they are not. See the INHIBIT=ERPW operand in the *NCP Generation and Utilities Guide.*

<u>NMONITOR</u> | **MONITOR** (Basic mode only)
Indicates whether or not VTAM is to invoke the ATTN exit routine (see EXLST macro) when this terminal causes an attention interruption. MONITOR means that the communications controller will monitor the terminal for attention interruptions while the terminal is not engaged in pending or actual I/O operations, and invoke the routine when an interruption is detected.

MONITOR is valid only if the user indicated during VTAM definition that the communications controller is to react to attention interruptions. If an attention interruption is received *during* an I/O operation, the I/O request ends with the RPL's FDBK2 field posted to indicate why. MONITOR does not apply to attention interruptions issued during an I/O operation.

If NMONITOR is specified, no monitoring occurs (the attention interruption is ignored).

NELC|**ELC** (Basic mode only)
Indicates whether communication control characters are to be generated for the data sent to this terminal communication. ELC signifies that the application program is embedding its own communication control characters in the data; its use prevents the system from doing so. NELC means that the application program is relying on the system to insert appropriate communication control characters. See Appendix B for a list of the communication control characters that are normally inserted. ELC can only be used if the NBINARY option code is in effect for the RPL. For basic-mode 3270 devices, NELC must be used.

NBINARY|BINARY (Basic mode only)
Indicates how data is to be handled when a WRITE macro instruction is used to write to a BSC device. When BINARY is specified, the data is sent in transparent text mode. This means that each of the communication control characters normally inserted by the communications controller is preceded by a DLE line control sequence. Any bit patterns can thus be sent, including communication control characters and object code. NBINARY means that the data is not sent in transparent text mode. Since the data will be screened for communication control characters, no bit patterns that happen to be communication control characters should be in the output data. (See "Transmission in Transparent Mode" in the *NCP Generation and Utilities Guide* for a description of transparent text mode.)

Example

```
NIB1        NIB        NAME=KBD3270,USERFLD=A(KBDRTN),
                       MODE=BASIC,LISTEND=YES
```

NIB1 could represent the keyboard component of a 3270 device whose entry in the resource definition table is labeled KBD3270. When OPNDST is issued to connect this terminal to the program, the NIB field of the OPNDST's RPL must point to NIB1. Since LISTEND=YES is coded, *only* this terminal can be connected with the OPNDST macro.

NIB Fields Not Set by the Application Program

All of the operands described above cause NIB fields to be set when the NIB macro instruction is assembled. There are additional NIB fields that can be examined by the application program during program execution, but cannot be set by the application program. VTAM uses these fields to return information to the application program upon completion of OPNDST processing:

Field Name	Content
CID	If the terminal named in the NIB is connected, this field contains a 32-bit value representing the symbolic name you supplied in the NAME field of the NIB. This shortened name, or CID, is also placed in the ARG field of the RPL used by the OPNDST macro instruction. Subsequent I/O requests for the terminal must have this CID in the ARG field of the RPL used for the I/O request. When NIB lists are used, the CID placed in the ARG field is not meaningful. The CID can be examined with the SHOWCB and TESTCB macro instructions. (If the terminal is not connected, the CID field is not modified.)

Field Name	Content
CON	An indicator that is set to show that the terminal represented by this NIB has been connected. You can examine this field following OPNDST by coding CON=YES in a TESTCB macro instruction; an "equal" PSW condition code indicates that the CON field is set to YES, and the terminal is connected. This field is useful if you are using OPNDST to establish connection with more than one terminal. Should connection be established with some of the terminals and not others, examination of each NIB's CON field will tell you which terminals are connected. This field must be reset before the NIB can be reused. (If the terminal is not connected, the CON field is not modified.)
DEVCHAR	An 8-byte field describing the type of terminal that has been connected and what optional features it has. (If the terminal is not connected, the DEVCHAR field is not modified.) This field can be examined with either the SHOWCB or TESTCB macro instruction or with the ISTDVCHR DSECT. The DEVCHAR DSECT (ISTDVCHR) is described in Appendix H (Figure H-13).

Devices Applicable for Each NIB Processing Option

Figure 8 shows, for each device supported by VTAM, the processing options applicable to that device. An X indicates that the PROC operand value is meaningful for the device.

	BLOCK	MSG	TRANS	CONT	LGIN–NLGIN	LGOUT–NLGOUT	CONFTXT–NCONFTXT	TMFLL–NTMFLL	EIB–NEIB	TIMEOUT–NTIMEOUT	ERPIN–NERPIN	ERPOUT–NERPOUT	MONITOR–NMONITOR	ELC–NELC	TRUNC–KEEP	BINARY–NBINARY	DFASYX–NDFASYX	RESPX–NRESPX	CA–CS–RPLC
Start–Stop Devices																			
IBM 1050 Data Communication System	X		X	X			X	X		X	X	X	X	X	X				X
IBM 2740 Communication Terminal, Model 1			X	X			X	X		X				X	X				X
IBM 2740 Communication Terminal, Model 1 with checking	X		X	X			X	X		X	X	X		X	X				X
IBM 2740 Communication Terminal, Model 1, with station control			X	X			X	X		X				X	X				X
IBM 2740 Communication Terminal, Model 1, with checking and station control	X		X	X			X	X		X	X	X		X	X				X
IBM 2740 Communication Terminal, Model 2			X	X			X				X	X			X				X
IBM 2741 Communication Terminal			X	X			X	X		X			X	X	X				X
IBM Communicating Magnetic Card Selectric Typewriter			X	X			X			X			X	X	X				X
IBM World Trade Telegraph Station			X	X			X							X	X				X
IBM SYSTEM/7			X	X			X			X				X	X				X
AT&T 83B3 Selective Calling Station			X	X			X							X	X				X
CPT-TWX, Models 33 & 35			X	X			X	X		X			X	X	X				X
Western Union Plan 115A Station			X	X			X							X	X				X
BSC and Local Non-SNA 3270 Devices																			
IBM 2770 Data Communication System	X	X	X	X			X		X		X	X		X	X	X			X
IBM 2780 Data Transmission Terminal	X	X	X	X			X		X		X	X		X	X	X			X
IBM 2972 General Banking Terminal, Models 8 and 11			X	X			X		X		X	X		X	X				X
IBM 3270 Information Display System, locally attached to controller (channel)			X				X								X				X
IBM 3270 Information Display System, remotely attached to controller (basic mode)			X				X								X				X
IBM 3270 Information Display System, locally or remotely attached, treated as an SNA device (record-mode)							X								X		X	X	X
IBM 3735 Programmable Buffered Terminal	X	X	X	X			X		X	X	X	X		X	X	X			X
IBM 3740 Data Entry System	X	X	X	X			X		X		X	X		X	X	X			X
IBM 3780 Data Transmission Terminal	X	X	X	X			X		X		X	X		X	X	X			X
IBM SYSTEM/3	X	X	X	X	X	X	X		X		X	X		X	X	X			X
IBM SYSTEM/370	X	X	X	X	X	X	X		X		X	X		X	X	X			X

Figure 8 (Part 1 of 2). Devices Applicable to Each NIB Processing Option

SNA Devices	BLOCK	MSG	TRANS	CONT	LGIN–NLGIN	LGOUT–NLGOUT	CONFTXT–NCONFTXT	TMFLL–NTMFLL	EIB–NEIB	TIMEOUT–NTIMEOUT	ERPIN–NERPIN	ERPOUT–NERPOUT	MONITOR–NMONITOR	ELC–NELC	TRUNC–KEEP	BINARY–NBINARY	DFASYX–NDFASYX	RESPX–NRESPX	CA–CS–RPLC
IBM 3270 Information Display System							X								X		X	X	X
IBM 3600 Finance Communications System							X								X		X	X	X
IBM 3650 Retail Store System *							X								X		X	X	X
IBM 3660 Supermarket System *							X								X		X	X	X
IBM 3767 Communication Terminal							X								X		X	X	X
IBM 3770 Data Communications System							X								X		X	X	X
IBM 3790 Communication System							X								X		X	X	X
IBM System/32 Batch Work Station							X								X		X	X	X

* Not available with OS/VS2 SVS

Figure 8 (Part 2 of 2). Devices Applicable to Each NIB Processing Option

OPEN–Open One or More ACBs

The OPEN macro instruction opens (or "activates") the ACB so that the ACB and all subsequent requests referring to it can be identified by VTAM as representing a specific application program. The programmer coding the OPEN macro instruction indicates the ACB (or ACBs) that are to be opened.

An ACB represents an application program, as defined by the installation. By means of an ACB's APPLID field the application program associates an ACB with an APPL entry. (A symbolic name is generated during VTAM definition by the installation when it defines the application program; the entry is generated with the APPL definition statement, and is called an APPL entry.) It is during OPEN processing that the association between the ACB and the APPL entry is actually made. One effect of this association is this: terminals directing logon requests to this APPL entry will in effect be directing their logon requests to the entry's associated ACB.

If you do not specify an APPL entry in the ACB's APPLID field, VTAM assumes that the APPL entry will be identified via JCL. In OS/VS, the name specified on the application program's EXEC statement is used; in DOS/VS, the job name is used. If you fail to indicate an APPL entry through either the APPLID field or by JCL, OPEN will not be completed successfully; and, in DOS/VS, if more than one ACB is opened by a single task, the OPEN will fail if more than one of the ACBs named has no name specified for APPLID. For this reason, the application program should always supply an APPL entry in the APPLID field for DOS/VS.

OPEN (and also CLOSE) must be issued in the mainline program (or in the LERAD or SYNAD exit-routines if they have been entered directly from the main program). Never issue OPEN in the RPL exit-routine or in any of the other exit-routines.

Privileged OS/VS2 MVS programs can issue OPEN macro instructions in any task, but all OPENs must be issued in the same task. In DOS/VS, OS/VS1 and OS/VS2 SVS, programs can issue OPEN macro instructions in more than one task.

If the APPL entry created by the installation contains a password, the ACB being opened must also specify that same password, or OPEN will not be completed successfully.

If the ACB being opened has its MACRF field set to LOGON, logon requests are queued for the application program. When SETLOGON (OPTCD=START) is subsequently issued, queued and new logon requests will cause the LOGON exit-routine to be scheduled.

Name	*Operation*	*Operands*
[symbol]	**OPEN**	**acb address[, acb address]...**
This form of OPEN is valid in DOS/VS only.		
[symbol]	**OPEN**	**(acb address[,,acb address]...)**
This form of OPEN is valid in OS/VS only.		

acb address

Indicates the ACB that is to be associated with an APPL entry.

Format: If more than one ACB is specified, separate each with a comma if the program is going to be run under DOS/VS. Separate each ACB address with *two* commas if the program is going to be run under OS/VS. (The same assembler handles both VTAM and non-VTAM macro instructions. An extra operand can be supplied with each address for the latter, and so an extra comma is required for the VTAM OPEN.)

Note: *VSAM ACB addresses can also be used in the OPEN macro instruction. DOS/VS users can also code DTF addresses, and OS/VS users can also code DCB addresses. The addresses of different types of control blocks can be combined in one OPEN macro instruction, although DOS/VS users are limited to a total of 15 addresses.*

Example

```
OPEN123    OPEN    ACB1,ACB2,(7)         (DOS/VS)

OPEN123    OPEN    (ACB,,ACB2,,(7))      (OS/VS)
```

OPEN123 opens ACB1, ACB2, and the ACB whose address is contained in register 7. Each of these ACBs is linked with an APPL entry in the resource definition table.

Return of Status Information

When control is returned to the instruction following the OPEN macro instruction, register 15 indicates whether or not the OPEN processing was completed successfully. Successful completion means that *all* ACBs specified in the OPEN macro instruction were opened; unsuccessful completion means that at least one ACB was not opened. Successful completion is indicated by a return code of 0 in register 15. For DOS/VS users, register 15 is unmodified if the OPEN is completed successfully and so should be set to 0 before OPEN is executed. Unsuccessful completion is indicated by the following register 15 values:

For DOS/VS	**Meaning**
Nonzero	One or more ACBs (or DTFs or VSAM ACBs) were not successfully opened.

For OS/VS	**Meaning**
4	All ACBs (or DCBs) were successfully opened, but warning messages were issued for one or more VSAM ACBs.
8	One or more ACBs (or DCBs) were not successfully opened. If the error condition indicated by the unopened ACB's ERROR field can be eliminated, another OPEN macro instruction can be issued for the unopened ACBs.
12	One or more ACBs (or DCBs) were not successfully opened. Another OPEN macro instruction cannot be issued for the unopened ACBs.

If unsuccessful completion is indicated, the application program should examine the OFLAGS field in each ACB to determine which one (or ones) could not be opened. Test each OFLAGS field by coding an ACB address and OFLAGS=OPEN in a TESTCB macro instruction; if the resulting PSW condition code indicates an equal comparison, that ACB has been opened:

```
TESTCB      ACB=ACB4,OFLAGS=OPEN
BE          OPENOK
```

If an unequal comparison is indicated, meaning that the ACB has not been opened, another field in that ACB can be checked to determine the reason. This field is the ERROR field. Like OFLAGS, ERROR is not a field that the application program should modify (that is, there is no ERROR operand for the ACB macro, and thus none for the MODCB macro), but the application program can obtain the contents of this field with the SHOWCB or TESTCB macro instruction. For example:

```
SHOWCB      ACB=ACB1,FIELDS=ERROR,AREA=SHOWIT,LENGTH=4,
            AM=VTAM
```

Note: *If the ACB is already open, or if the address specified in the OPEN macro instruction either does not indicate an ACB or lies beyond the addressable range of your application program,* nothing *is posted in the ACB's ERROR field. Thus if you find one of the following return codes in the ACB's ERROR field and none of the specified causes apply, perhaps you are actually examining a field whose contents have not been modified by OPEN. An already-open ACB or an invalid ACB address will result in register 15's being set to a non-zero value, however.*

These are the values that can be set in the ERROR field of an ACB (except where noted, all values apply to DOS/VS and to OS/VS):

Value Hex (Decimal)	Meaning
0 (0)	OPEN has successfully opened this ACB.
14 (20)	OPEN cannot be processed because of a temporary shortage of storage. (OS/VS only)
24 (36)	The password specified by the ACB does not match the password specified in the corresponding APPL entry, or the ACB does not specify a password and one was specified in the APPL entry.
46 (70)	OPEN was issued in an exit routine.
50 (80)	VTAM has not been included as part of the operating system. The fault lies in your installation's system definition procedures.
52 (82)	VTAM has been included as part of the operating system, but the network operator has issued a HALT command, and VTAM is shutting down. You cannot attempt connection or communication with any terminals.
54 (84)	Either the address supplied in the ACB's APPLID field lies beyond the addressable range of your application program, or no entry could be found in the resource definition table that matches the name indicated by the ACB's APPLID field (or supplied by JCL). If the OPEN macro instruction was specified correctly, your installation may have failed to include your application program's

symbolic name during VTAM definition, or you may have improperly handled the symbolic name. Refer to the description of the APPLID operand in the ACB macro instruction.

56 (86) A match for your application program's symbolic name was found, but it was for an entry other than an APPL entry. If you specified this name in the ACB's APPLID field, verify that the installation has supplied you with the correct name and that you have handled this name properly (see the APPLID operand of the ACB macro instruction). If the symbolic name was supplied via JCL, the job step name, job name, or task ID is suspect.

58 (88) Another ACB, already opened by VTAM, indicates the same application program symbolic name that this ACB does. The installation may have assigned the same symbolic name to two application programs. This is valid only if the programs do not run (or are at least not open) concurrently. Possibly the system operator initiated your job or job step at the wrong time.

5A (90) No entry could be found in the resource definition table that matches the name indicated by the ACB's APPLID field (or supplied via JCL). This error may have occurred because the installation deactivated the APPL entry or never created it.

5C (92) Either VTAM has been included as part of the operating system, but is inactive, or (for DOS/VS only) the priority of the application program is greater than VTAM's priority. VTAM must have the higher priority.

5E (94) The address supplied in the ACB's APPLID field lies beyond the addressable range of your application program.

60 (96) An apparent system error occurred. Either there is a defect in VTAM's logic, or there is an error in your use of OPEN that VTAM did not properly detect. Save all applicable program listings and storage dumps, and consult your IBM Programming Services Representative.

62 (98) The APPLID length indicator byte is incorrectly specified.

64 (100) The address supplied in the ACB's PASSWD field lies beyond the addressable range of your application program.

66 (102) The PASSWD length indicator byte is incorrectly specified.

68 (104) The APPLID field in the ACB identifies an application program that is defined with AUTH=PPO in its APPL statement. Another program with the same authorization is already active. Only one program defined with AUTH=PPO may be active at a time.

70 (112) You attempted to open an ACB that is in the process of being closed. This can occur when a VTAM application program job step or subtask is canceled or terminates abnormally. The process of closing the ACB may continue after the job step or subtask has actually terminated. Subsequently, if the job step is restarted or

the subtask is reattached before the ACB closing process has been completed, an OPEN macro that is then issued for that ACB will fail. (OS/VS only)

88 (136) OPEN cannot be processed because of a temporary storage shortage. (DOS/VS only)

Since most of the error conditions described above result from an error in your application program or in the installation's definition of VTAM, there is little that can be done during program execution when these return codes are encountered. If, however, you are attempting to open more than one ACB, you may wish to check the ERROR field of each ACB. All ACBs whose ERROR fields are set to zero have been opened successfully, and your application program can proceed using those ACBs.

Although the return codes following a DOS/VS OPEN are not identical to those following an OS/VS OPEN, the following procedure can be used to produce a system-independent determination of a successful or unsuccessful OPEN:

- Zero register 15 before issuing OPEN.
- Issue OPEN for only one VTAM ACB at a time.
- If register 15 is zero, consider the OPEN successful.
- If register 15 is nonzero, consider the OPEN unsuccessful. Examine the contents of the ACB's ERROR field.

OPNDST–Establish Connection with Terminals

The OPNDST (open destination) macro instruction requests VTAM to establish connection between the application program and one or more terminals (logical units, BSC or start-stop terminals, or local non-SNA 3270s).

Connection must be established with a terminal before the application program can communicate with that terminal. OPNDST is the sole means by which this connection can be requested. There are, however, two fundamentally different ways that OPNDST can be used to request connection.

Acquiring a Terminal

An application program can initiate a request that a terminal be connected to it. This process is called acquiring a terminal. Such a request is satisfied if the terminal is available; that is, is not connected to another application program and does not have a pending logon. This type of request is implemented by setting the ACQUIRE option code in the RPL used by OPNDST. The use of ACQUIRE must be authorized by the user.

If a BSC or start-stop terminal has been defined as dial-out by the user, the connection request is completed immediately if the terminal is available, but the terminal is dialed only when an I/O request is issued for the terminal. If a logical unit has been defined as dial out, it is dialed when OPNDST is issued for it.

Accepting a Terminal

In the second way of using OPNDST, the application can request that a terminal be connected to it only if the terminal requests connection with that application program or if a connection request is made on its behalf. This process is called accepting a terminal.

This type of connection request can be embedded in a LOGON exit routine (see the EXLST macro) that is automatically entered when a terminal logs on. This arrangement means that terminal logon requests can, in effect, invoke the type of OPNDST which will complete the connection.

This type of request is implemented by setting the ACCEPT option code in the RPL.

Note: *When a terminal logs on, VTAM first checks for outstanding OPNDST requests (that is, OPNDSTs with ACCEPT and Q that have not yet been completed because no logon has been made by the terminal operator). If there are no outstanding OPNDST requests, VTAM checks for a LOGON exit routine and schedules it if it is active. Thus a logon will not cause a LOGON exit routine to be scheduled if there is a pending OPNDST with OPTCD=ACCEPT. If no LOGON exit routine exists, the logon is queued for an eventual OPNDST ACCEPT request. See the* Macro Language Guide *for more information.*

There are three versions of OPNDST with OPTCD=ACQUIRE. These versions are specified with two RPL option codes, CONALL and CONANY, and the LISTEND field of the NIB, which govern whether multiple terminals, only one terminal, or one specific terminal is to be connected. OPNDST with ACCEPT likewise has two versions, specified with two more RPL option codes, SPEC and ANY. These govern whether a specific terminal or any eligible terminal is to be connected. There are therefore a total of five general types of OPNDST. The following five sections indicate how you must prepare for each and what happens when the connection occurs.

OPNDST with ACQUIRE, CONALL, and a NIB List: Set the RPL's NIB field to point to a *list* of NIBs (described in the LISTEND operand of the NIB macro instruction), and set the ACQUIRE and CONALL option codes in the RPL.

When OPNDST is issued, VTAM connects the program to *all* of the terminals represented in the NIB list that are available when OPNDST is executed. VTAM also:

- Generates a CID for each connected terminal. Each CID is placed in its respective NIB in the CID field, where it can be obtained with the SHOWCB macro instruction when it is needed. The CID is not placed in the RPL's ARG field. (In the other forms of OPNDST, the CID is placed in the NIB *and* in the RPL.)
- Sets a flag in each NIB indicating that the terminal is connected. This flag can be tested by specifying CON=YES in a TESTCB macro instruction.
- Places the address of the first NIB of the NIB list in the AREA field of the RPL. (This is the same address you supplied in the RPL's NIB field; it is returned to you because the contents of the NIB field are modified by VTAM during connection.)

Caution must be taken when using this type of OPNDST. If all terminals are not active (that is, they have not been activated by a VARY ACT network operator command), the OPNDST will fail and none of the terminals will be connected.

OPNDST with ACQUIRE, CONANY, and a NIB List: Set the RPL's NIB field to point to a *list* of NIBs, and set the ACQUIRE and CONANY option codes in the RPL.

When OPNDST is issued, VTAM connects the program to the *first* (and only to the first) available terminal represented in the NIB list. If no terminals are available, VTAM terminates the OPNDST request. A terminal is available if it is not connected to any application program and has not been logged on to an application program. VTAM also:

- Generates a CID for the connected terminal and places it in the ARG field of the RPL and in the CID field of the NIB.
- Sets a flag in the NIB that represents the connected terminal. The application program can locate this NIB by issuing a TESTCB macro instruction with the CON=YES operand for each NIB. An "equal" PSW condition code is set following the TESTCB macro instruction if the NIB being tested represents the connected terminal.
- Places the address of the first NIB of the NIB list in the AREA field of the RPL. (This is the same address you supplied in the RPL's NIB field; it is returned to you because the contents of the NIB field are modified by VTAM during connection.)

OPNDST with ACQUIRE and One NIB: Set the RPL's NIB field to point to *one* NIB (LISTEND field set to YES). VTAM connects the terminal whose symbolic name is in the NIB to the application program. VTAM completes the request immediately; if the terminal is not immediately available, it is not connected. VTAM also:

- Generates a CID for the connected terminal and places it in the ARG field of the RPL and in the CID field of the NIB.
- Sets a flag in the NIB indicating that the terminal is connected.
- Places the address of the NIB in the AREA field of the RPL.

OPNDST with ACCEPT and ANY: Set the RPL's OPTCD field to ACCEPT and ANY, and set the NIB field to point to a NIB. This NIB need not have any symbolic name in it, but it must at least have the MODE field set to RECORD or BASIC and the LISTEND field set to YES.

When OPNDST is issued, VTAM attempts to connect the program to any terminal that has directed a logon to the program. VTAM also:

- Places the symbolic name of the connected terminal into the NAME field of the NIB.
- Generates a CID for the connected terminal and places it in the CID field of that NIB and in the ARG field of the RPL.
- Places the address of the NIB in the RPL's AREA field.

Note: *In a network that contains SNA and non-SNA terminals, the mode set in the NIB may not apply to the terminal that logs on (for example, the MODE operand is set to BASIC and a logical unit is the first terminal to log on). If a LOGON exit routine is used, the application program can issue an INQUIRE macro instruction to determine how the MODE field should be set.*

OPNDST with ACCEPT and SPEC: Set the RPL's OPTCD field to ACCEPT and SPEC, and set the NIB field to point to a NIB. The NAME field of this NIB must identify the terminal and have its LISTEND field set to YES.

When OPNDST is issued, the result will be this: VTAM connects the program to the specific terminal represented in the NIB if (or when) that terminal has directed a logon to the program. VTAM also:

- Generates a CID for the connected terminal and places it in the CID field of the NIB and in the ARG field of the RPL.
- Places the address of the NIB in the RPL's AREA field.

Besides the SPEC-ANY option code, other RPL and NIB options and fields affect how the OPNDST request is handled. Generally, their effect is the same as it is for other macro instructions that point to an RPL; see Figure 10 in the RPL macro instruction description for a list of these codes and fields.

Name	*Operation*	*Operands*
[symbol]	**OPNDST**	**RPL=rpl address** **[,rpl field name=new value] . . .**

RPL=rpl address

Indicates the location of the RPL to be used during OPNDST processing.

rpl field name=new value

Indicates an RPL field to be modified and the new value that is to be contained within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the OPNDST macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field to be modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register.

The following RPL operands apply to an OPNDST macro instruction:

ACB=acb address
Indicates the ACB that identifies the application program to which the terminal is to be connected.

NIB=nib address
Indicates the NIB whose PROC, MODE, and USERFLD attributes are to be assigned to the connected terminal. The LOGMODE and, optionally, the BNDAREA field of the NIB specify the session parameters to be used in creating an SNA Bind command. If OPTCD=ACQUIRE or OPTCD=SPEC, the NIB also identifies (via its NAME field) the terminal to be connected.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) OPNDST macro instruction is completed. The macro instruction is completed when the connections between the application program and the terminals specified are established. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When the SYN option code is set, control is returned to the application program when the macro instruction has been completed. When ASY is set, control is returned as soon as VTAM has accepted the request. Once one or more connections have been established (that is, once the macro instruction has been completed), the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB-EXIT field.

OPTCD=ACQUIRE | ACCEPT
When ACQUIRE is set, VTAM connects the terminal or terminals indicated via the RPL's NIB field. Only available terminals are connected. When ACCEPT is set, a terminal for which there is a logon for the application program is connected.

OPTCD=CONANY | CONALL
When CONANY is set and OPNDST (OPTCD=ACQUIRE) is issued, connection is made to the first available terminal of the NIB list indicated in the RPL's NIB field. When CONALL is set, connection is made to *all* the available terminals in the list.

OPTCD=SPEC | ANY
When SPEC is set, the terminal identified by the NIB's NAME field is connected if and when that terminal directs a logon to the application program. When ANY is set, *any* terminal that has issued a logon for the application program is connected.

OPTCD=CS | CA
Specifies the initial setting of the connected terminals' CS-CA mode. When CA is set, data obtained from a terminal can satisfy a READ or RECEIVE (OPTCD=ANY) macro instruction. When CS is set, only READ or RECEIVE (OPTCD=SPEC) macro instructions can obtain data from a terminal.

OPTCD=Q | NQ
When Q is set, VTAM connects the terminal when it becomes available, no matter how long that might take. When NQ is set, VTAM terminates the OPNDST macro instruction immediately if the terminal is not immediately available. This option applies only when OPTCD=ACCEPT is in effect; when OPTCD=ACQUIRE is in effect, this option is ignored.

Examples

Note: *To avoid obscuring the differences between the basic types of OPNDST, the same technique is used to set the RPL fields in each example (namely, inserting RPL-modifiers on the OPNDST macro instruction). RPL fields could just as well have been set with the MODCB macro instruction, with assembler instructions, or with the RPL macro instruction itself.*

This is an "ACQUIRE CONALL" OPNDST

```
ACQALL   OPNDST  RPL=RPL1,NIB=NIBLST1,ACB=ACB1,
                 OPTCD=(ACQUIRE,CONALL)
         .
         .
         .
NIBLST1  NIB     NAME=TERM1,MODE=BASIC,LISTEND=NO
         NIB     NAME=TERM2,MODE=BASIC,LISTEND=NO
         NIB     NAME=TERM3,MODE=BASIC,LISTEND=YES
```

ACQALL connects all of the available terminals of NIBLST1 (TERM1, TERM2, and TERM3) to the application program represented by ACB1.

This is an "ACQUIRE CONANY" OPNDST

```
ACQANY   OPNDST  RPL=RPL2,NIB=NIBLST2,ACB=ACB1,
                 OPTCD=(ACQUIRE,CONANY)
         .
         .
         .
NIBLST2  NIB     NAME=LUX,MODE=RECORD,LISTEND=NO
         NIB     NAME=LUY,MODE=RECORD,LISTEND=NO
         NIB     NAME=LUZ,MODE=RECORD,LISTEND=YES
```

ACQANY connects *one* of the logical units of NIBLST2 (LUX, LUY, or LUZ) to the application program. The CON and CID fields are set in the NIB containing the name of the connected logical unit. RPL's ARG field also contains the CID of the connected logical unit.

This is an "ACQUIRE One NIB" OPNDST

```
ACQONE   OPNDST   RPL=RPL3,NIB=NIB3,ACB=ACB1,
                  OPTCD=ACQUIRE
            .
            .
            .
NIB3     NIB      NAME=TERM35,MODE=BASIC
```

ACQONE connects TERM35 to the application program if TERM35 is available.

This is an "ACCEPT ANY" OPNDST

```
ACPTANY  OPNDST   RPL=RPL4,NIB=NIB6,ACB=ACB1,
                  OPTCD=(ACCEPT,ANY,NQ)
            .
            .
            .
NIB6     NIB      MODE=RECORD
```

ACPTANY connects any one logical unit that has issued a logon to the application program. The symbolic name of this logical unit (along with its CID) is placed in NIB6. Since NQ is specified, the request will be terminated if no logical unit has issued a logon.

This is an "ACCEPT SPEC" OPNDST

```
ACPTSPC  OPNDST   RPL=RPL5,NIB=NIB7,ACB=ACB1,
                  OPTCD=(ACCEPT,SPEC,Q)
            .
            .
            .
NIB7     NIB      NAME=LU77,MODE=RECORD
```

ACPTSPC connects LU77 to the application program when a logon is queued from the logical unit to the application program.

Return of Status Information

After the OPNDST operation is completed, the following NIB fields are set:

The CON field is set to YES if the terminal was in fact connected; otherwise this field is not modified. If it is set, the CON field must be reset before the NIB can be reused. This field can be examined by coding CON=YES on a TESTCB macro instruction.

If the terminal was connected, the connected terminal's CID is placed in the CID field. (See the note.)

If the ACCEPT and ANY options were in effect, the symbolic name of the connected terminal is placed in the NAME field.

The characteristics of the connected terminal are indicated in the DEVCHAR field. The DEVCHAR codes are explained in Appendix H.

The following fields are set in the RPL:

If one (and only one) terminal has been connected, the CID of the connected terminal is placed in the ARG field. (See the note.)

The address of the NIB or NIB list (as supplied by you in the NIB field) is returned in the AREA field. The NIB field is overlayed when the CID is placed in the ARG field, since the NIB and ARG fields occupy the same physical location in the RPL.

The value 23 (decimal) is set in the REQ field, indicating an OPNDST request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C. If the return codes indicate that the OPNDST has failed, do not issue a CLSDST for the terminal. (See the note.)

The SSENSEI, SSENSMI, and USENSEI fields are set if RTNCD=16 and FDBK2=1 are set in the RPL (OPNDST for a logical unit failed).

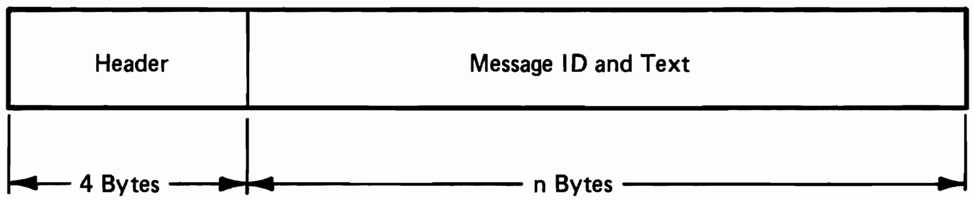
Note: *Even if the RTNCD and FDBK2 fields of the RPL indicate a successful OPNDST completion, the CON field of each NIB must still be examined to determine if the terminal is connected. If the CON field is still not set to YES, this indicates that a CLSDST has already been done, or initiated for this terminal and this terminal is not connected.*

Registers 0 and 15 are also set as indicated in Appendix C. (Note that the USERFLD field is not set for OPNDST.)

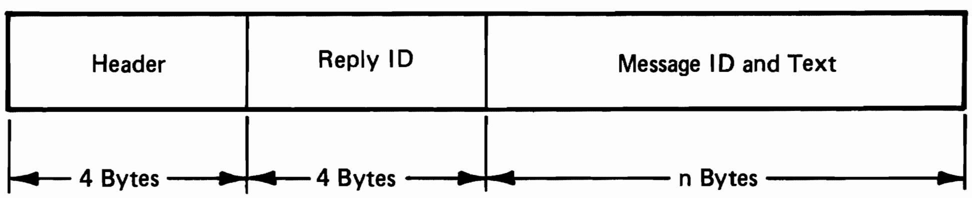
RCVCMD–Receive a Message from VTAM

An application program uses the RCVCMD macro instruction to receive VTAM messages that were generated by VTAM when processing a network operator command that it issued by a SENDCMD macro instruction. In addition, unsolicited VTAM messages such as those indicating an unexpected failure in the network can be received with this macro instruction.

The RCVCMD macro instruction points to an RPL whose AREA field points to the location in the application program that is to receive the message. Every message received from VTAM consists of a header followed by the message ID (ISTxxxx for OS/VS or 5xxxx for DOS/VS) and the text. The general format of the header and message is:



If a message requires a reply, an additional reply ID is inserted between the header and the message ID. A REPLY command can then be returned to VTAM using the SENDCMD macro instruction (see the description of the SENDCMD macro instruction for more information). The general form of the header, reply ID, and message is:



For information on using the data area and writing an application program that can issue network operator commands and receive VTAM messages, see the publication, *Supplement to the VTAM Macro Language Guide for the Program Operator,* GC27-0036.

The use of the RCVCMD macro instruction must be authorized when the application program is defined to VTAM. There are two levels of authorization available, primary and secondary. An application program that is designated as a primary program may receive responses to VTAM network operator commands issued using SENDCMD and unsolicited VTAM messages. An application program designated as a secondary program can receive only responses to network operator commands that it issued. If a primary program is not active, unsolicited VTAM messages are sent to the system console.

Name	Operation	Operands
[symbol]	RCVCMD	RPL=rpl address [,rpl field name=new value] . . .

RPL=rpl address

Indicates the location of the RPL that describes the RCVCMD operation.

rpl field name=new value

Indicates an RPL field name to be modified and the new value that is to be contained or represented within it. To avoid the possibility of program reassembly following some future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the RCVCMD macro instruction.

Format: For *rpl field name,* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction.

The following RPL operands apply to the RCVCMD macro instruction:

ACB=acb address

Indicates the ACB that identifies the application program.

AREA=message address

Must contain the address of the area in the application program where the incoming header, optional message identification, and the message text are to be placed. After the message has been moved to this area, the RPL's RECLEN field is set by VTAM with the total number of bytes of received data. The AREA field is ignored if AREALEN=0.

AREALEN=length of message area

Contains the length (in bytes) of the message area pointed to by AREA. The length specified should be 8 bytes longer than the longest message anticipated to account for the message header and optional reply ID. The AREA field should be no less than 4 bytes and no longer than 130 bytes.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address

Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) RCVCMD operation is completed. The RCVCMD request is completed when the message or reply has been received, the data (if any) has been placed in the input area, and the appropriate information has been set in the RPL. If NQ is specified and no input is available, RCVCMD is completed immediately. If EXIT is specified, the RPL exit routine is scheduled. Otherwise, the ECB is posted and CHECK or WAIT must be used to determine when posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
SYN specifies that control is returned to the application program when the RCVCMD operation is completed. ASY specifies that control is returned as soon as VTAM has accepted the RCVCMD request; once the operation has been completed, the ECB is posted or the RPL exit routine is scheduled as indicated by the ECB-EXIT field. See the RPL macro instruction for more information.

OPTCD=Q | NQ
Indicates the action to be taken if no input is available when the RCVCMD macro instruction is executed. OPTCD=Q means that the macro instruction is to be completed when the requested input eventually arrives. OPTCD=NQ means that the macro instruction is to be completed immediately with RTNCD=0 and FDBK2=6 if the input is not available.

OPTCD=TRUNC
Indicates that overlength input data is truncated whenever the RCVCMD macro instruction is issued. It is advisable to specify this option since overlength data is always truncated.

Note: *After a CLOSE macro instruction has been issued and messages are still queued for the application program, RCVCMD macro instructions may still be issued but they will not be queued. Therefore, these RCVCMD macro instructions must be issued with OPTCD=NQ. After the last message has been received, the return code for RCVCMD (RTNCD=0 and FDBK2=6) will indicate that no more messages are queued.*

Example

```
RCVCMD1    RCVCMD    RPL=RPL1,AREA=MSGBUF,AREALEN=126,
                     OPTCD=(TRUNC,Q)
```

RCVCMD1 is completed when an incoming message is received from VTAM. After RCVCMD1 is completed, the application program can examine the contents of MSGBUF to determine the message received. Any messages that exceed 126 bytes are truncated to 126 bytes.

Return of Status Information

After the RCVCMD operation is completed, the following RPL fields may be set by VTAM:

The RECLEN field indicates the length of the message placed in the input area pointed to by the AREA field. The length specified includes 4 bytes for the header and 4 additional bytes (if required) for the reply identifier. The reply-requested bits in the status field of the header can be tested to determine if this field is present.

The value 40 (decimal) is set in the REQ field, indicating a RCVCMD request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

READ–Read Data into Program Storage (Basic Mode Only)

The READ macro instruction obtains data from VTAM buffers and moves it into a designated area in program storage. It may or may not cause physical I/O to be performed. If OPTCD=ANY is in effect, the READ operation involves no I/O operation, but simply moves data already obtained from a terminal into program storage.

If READ is being used to obtain data from a specific BSC or start-stop terminal which means that the SPEC option code is in effect in the RPL–and no data from that terminal is available, READ first causes data to be solicited. This implied solicit operation works in the same manner as the solicit operation explained in the SOLICIT macro instruction description. (For 3270 terminals, this use of READ applies only for the first READ; the data for subsequent read operations is unsolicited.)

As soon as VTAM has moved the data into program storage, it sets the RPL's RECLEN field to indicate how many bytes of data were moved.

If the return code posted in register 15 indicates that the read operation was completed successfully, the application program should check the RPL's FDBK field to determine whether the data received represents the end of a message or transmission. (The read operation may obtain a block of data ending with an end-of-transmission indicator, or the indicator may come separately with the next read operation. In the latter case, the RECLEN field is set to 0 when the next read operation is completed.)

The user of the READ macro instruction codes the address of the RPL that will govern the read operation. Various fields in the RPL determine from which terminal the data is to be obtained, the location of the area in the program where the data is to be placed, and other information regarding how the read request is to be handled. The RPL fields can be modified with the READ macro instruction itself. The operands used to set these fields are indicated below.

The TRUNC-KEEP option determines how excess data is to be handled. When TRUNC is in effect and there is too much incoming data to fit in the input area, the data is truncated and the excess is lost. If KEEP is in effect instead, and there is too much data, the excess if held for the time being and moved into the storage area when the next READ is issued. Flags set in the RPL's FDBK field (explained in Appendix C) indicate when the last of the excess data has been read.

Name	*Operation*	*Operands*
[symbol]	**READ**	**RPL=rpl address** **[,rpl field name=new value] . . .**

RPL=rpl address
Indicates the location of the RPL that governs the read operation.

rpl field name=new value
Indicates an RPL field to be modified and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly

following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the READ macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. ARG can also be coded. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register. The value supplied for the ARG keyword must indicate a register.

The following RPL operands apply to a READ macro instruction:

ACB=acb address
Indicates the ACB that identifies the application program.

ARG=(register)
If data is to be read from a specific terminal, the ARG field of the RPL must contain the CID of that terminal. Register notation is required if the CID is to be placed in the ARG field with this READ macro instruction.

If data is to be read from *any* terminal, the ARG field's content when the macro is issued is irrelevant. After the data has been read, VTAM obtains the CID of the terminal from which the data originated and places it in the ARG field.

AREA=input data area address
The AREA field must contain the address of the area in the program where the data is to be placed. Once the data has been moved, the RPL's RECLEN field is posted by VTAM with the number of bytes that were placed there.

AREALEN=length of input data area
The AREALEN field must contain the length (in bytes) of the data area pointed to by AREA. This value is used by VTAM to determine whether there is too much incoming data to fit. If there is too much, the action indicated by the TRUNC-KEEP processing option is taken.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) READ macro instruction is completed. The macro instruction is completed after the input data has been moved into the application program's storage area. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT is required to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When the SYN option code is set, control is returned to the application program when the READ macro instruction has been completed. When ASY is set, control is returned as soon as VTAM has accepted the request. Once the macro instruction has been completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB-EXIT field.

OPTCD=SPEC | ANY
When the SPEC option code is set, data is obtained from the specific terminal identified in the ARG field and placed in program storage. If no previously solicited data from that terminal is being held in VTAM buffers, a solicit operation is performed and the data is moved into program storage. If data is available in VTAM buffers, the READ macro instruction merely moves the data from the buffers to program storage.

When ANY is set, only data already available from a terminal is moved to program storage. The user does not identify a terminal; the data can originate from *any* terminal connected to the application program. VTAM obtains the CID of the terminal from which the data originated and places it in the ARG field of the RPL.

A READ macro instruction with OPTCD=ANY can be issued when no terminals are connected if the application program has opened an ACB. The read request is queued until one or more terminals are connected and data arrives from any of them. If a read request with OPTCD=ANY has not been completed and the application program issues a CLSDST macro instruction disconnecting all of its terminals but does not close the ACB, the read request does not have to be reissued after a subsequent OPNDST is issued.

OPTCD=CA | CS
When the CA option code is set, there is no restriction on subsequent retrieval of data from the terminal that is the object of this READ macro instruction.

When CS is set, however, any subsequent input operation will exclude that terminal from the group of terminals eligible for input operations. This exclusion applies only if the ANY option code is in effect for the subsequent operation. See the RPL macro instruction for more information.

Examples

```
READ1     READ      RPL=RPL1,AREA=INFO,AREALEN=132,
                    OPTCD=(ANY,SYN)
            .
            .
            .
INFO      DS        CL132
```

READ1 scans VTAM buffers for data previously obtained from any connected terminal; and if none has yet been obtained, waits until data arrives. READ1 then places the data into INFO. The CID of the terminal from which the data originated is placed into the ARG field of RPL1. Control is not returned to the program until the read operation has been completed.

```
READ2     READ      RPL=RPL1,ARG=(3),AREA=INFO,AREALEN=132,
                    OPTCD=(SPEC,SYN)
            .
            .
            .
NIB1      NIB       NAME=TERM1,MODE=BASIC
INFO      DS        CL132
```

READ2 operates much like READ1 except that data is being read from a specific terminal. When the terminal was originally connected, the CID for that terminal was placed both in NIB1 and in the RPL used for the connection macro (OPNDST). This example assumes that the CID will be in register 3 when READ2 is executed.

Return of Status Information

Once the READ operation is completed, the following RPL fields are set:

The RECLEN field contains the number of bytes of data that were placed in the input area.

The ARG field contains the CID of the terminal from which the data originated.

The USER field is set. When a NIB is established, the user has the option of specifying any value in the USERFLD field of that NIB. When the READ macro instruction is subsequently issued for the terminal associated with that NIB, whatever had been placed in USERFLD by the user is placed in the USER field of the RPL by VTAM.

The value 29 (decimal) is set in the REQ field, indicating a READ request.

If READ is completed normally, as indicated by register 15 and the RTNCD field, the FDBK field is set indicating various attributes of the data just read. See Appendix C.

The SENSE field is set as indicated in Appendix C.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

RECEIVE–Receive Input from a Logical Unit (Record Mode Only)

The RECEIVE macro instruction moves a message from a VTAM buffer to an input area in the application program or sets RPL fields to indicate data flow control commands or a response. The data, data flow control commands, or responses were previously sent from a logical unit or a BSC or local non-SNA 3270 terminal used in record mode. If data is received, it is placed in the input area designated by the application program. If a response or control commands are received, various RPL fields are set accordingly. Figure 9 illustrates the major options for a RECEIVE macro instruction.

A RECEIVE macro instruction can obtain any one of the following types of input (when the RECEIVE is issued, the application program designates the type or types that can satisfy the macro instruction):

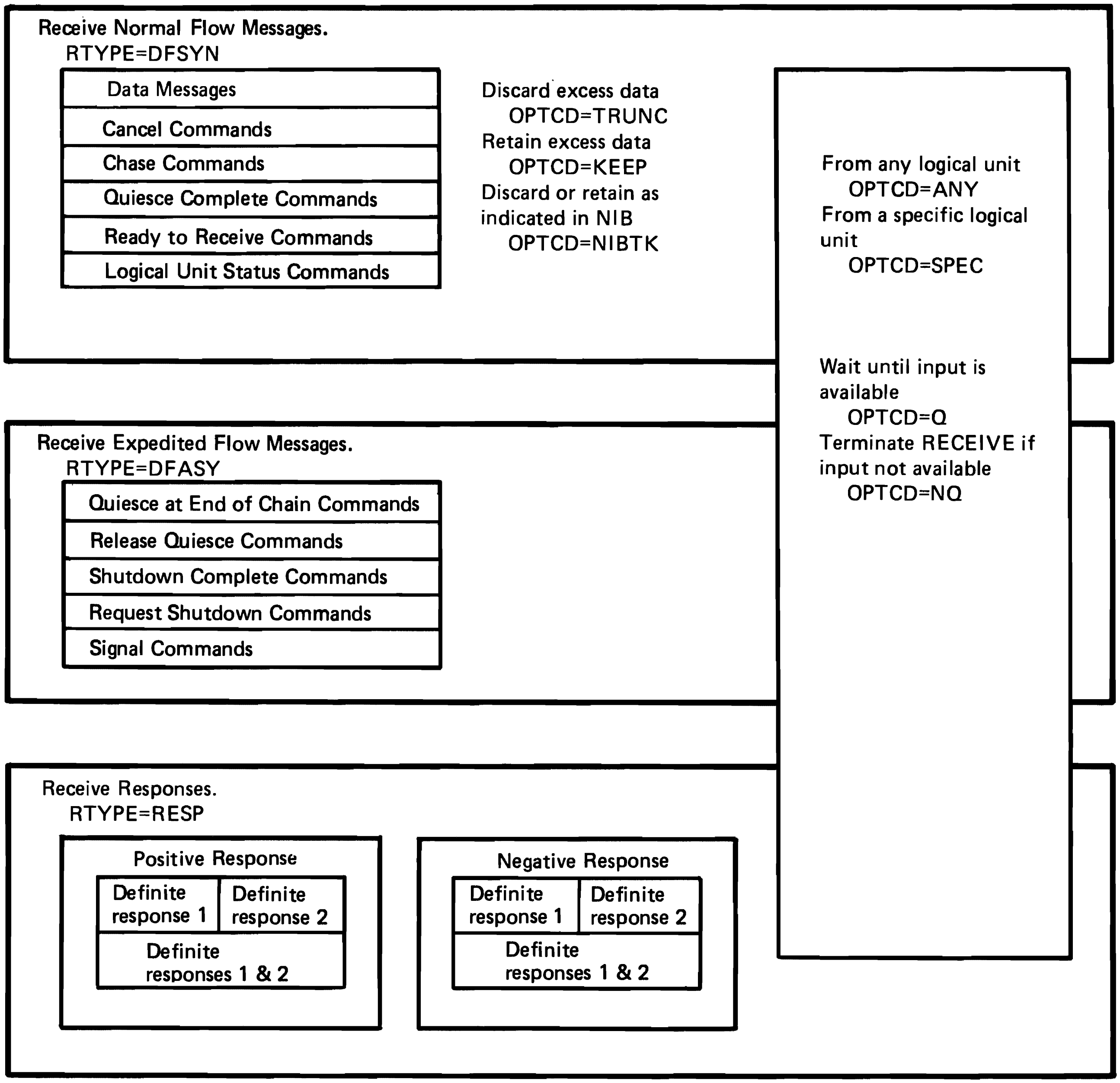


Figure 9. The Major RECEIVE Options

The application program designates which of the following types of input can cause the RECEIVE macro instruction to be completed (any combination can be selected):

- Normal-flow messages (such as data messages or Ready to Receive, Chase, or Cancel commands). Input from a 3270 for which MODE=RECORD was specified is included in this category.
- Expedited-flow messages (such as Release Quiesce, Quiesce at End of Chain, or Signal commands).
- Responses to data messages.

Only one type of input can satisfy the RECEIVE macro instruction. When the macro instruction is completed, the RPL's RTYPE field indicates the type actually received. If more than one type of input is available to satisfy a RECEIVE, the following priorities determine which type of input will satisfy the RECEIVE:

1. Expedited-flow messages
2. Responses
3. Normal-flow messages

Note: *When a logical unit sends a series of requests and responses to an application program, all of the requests and all of the responses will arrive in the order in which they were sent; however, they may not be intermixed in the same way. For example, if the logical unit sent the following:*

Response 1

Request 1

Request 2

Request 3

Response 2

Response 3

Request 4

They could arrive at the application program as follows:

Response 1

Response 2

Request 1

Response 3

Request 2

Request 3

Request 4

Name	*Operation*	*Operands*
[symbol]	**RECEIVE**	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address
Indicates the location of the RPL that describes the RECEIVE operation.

rpl field name=new value
Indicates an RPL field to be modified and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with this RECEIVE macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. ARG can also be coded. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or register notation can be used.

The following RPL operands apply to the RECEIVE macro instruction:

ACB=acb address
Indicates the ACB that identifies the application program and through which the sending logical unit was connected.

ARG=(register)
If a specific terminal is to be read (OPTCD=SPEC), the ARG operand specifies the register containing the CID of that logical unit. If the ARG field is not modified, the CID already in the RPL's ARG field is used.

AREA=input data area address
The AREA field must contain the address of the area in the application program where the incoming data is to be placed. If a command is received instead of data, the CONTROL field is posted with a value other than CONTROL=DATA, and the input data area is not used. Once the data has been moved, the RPL's RECLEN field is set by VTAM with the total number of bytes of received data. The AREA field is ignored if AREALEN=0.

AREALEN=length of input data area
The AREALEN field contains the length (in bytes) of the data area pointed to by AREA. This value is used by VTAM to determine if there is too much incoming data to fit. If there is too much, the action indicated by the TRUNC-KEEP-NIBTK option code is taken.

AREALEN=0 with OPTCD=KEEP can be used to determine the amount of incoming data (the total length is set in RECLEN). A data area can be obtained and the RECEIVE macro instruction reissued. AREALEN=0 with OPTCD=TRUNC can be used to eliminate unwanted data messages that are queued for the application program.

BRANCH=YES | NO
If RECEIVE is to be issued in an application program that is running in privileged state under a TCB (OS/VS2 MVS only), BRANCH can be set to YES. See the RPL macro instruction for more information.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous RECEIVE request (OPTCD=ASY) is completed. A RECEIVE request is completed when the message or response has been received, the data (if any) has been placed in the

input data area, and the appropriate information has been set in the RPL. If NQ is specified and no input is available, RECEIVE is completed immediately. If EXIT is specified, the RPL exit routine is scheduled. Otherwise, the ECB is posted and CHECK or WAIT must be used to determine when posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When SYN is set, control is returned to the application program when the RECEIVE operation is completed. When ASY is set, control is returned as soon as VTAM has accepted the RECEIVE request; once the operation has been completed, the ECB is posted or the RPL exit routine is scheduled as indicated by the ECB-EXIT field. See the RPL macro instruction for more information.

OPTCD=CA | CS
When the RECEIVE operation is completed successfully, the logical unit is placed into continue-any mode (CA) or into continue-specific mode (CS). This mode determines whether the next RECEIVE (OPTCD=ANY) can be satisfied by the logical unit's next transmission.

This option code has no effect if OPTCD=NQ and the RECEIVE is completed with no input.

When a RECEIVE macro instruction is completed, the CA/CS setting indicates how the application program must receive the next input of the type indicated in the RTYPE field. For example, if a RECEIVE macro instruction completes with OPTCD=CS and RTYPE=RESP, the application program must issue a RECEIVE RTYPE=RESP and OPTCD=SPEC to receive another response from the logical unit. However, at the same time, the application program could issue a RECEIVE RTYPE=DFSYN and OPTCD=ANY to receive a normal-flow request from the same logical unit (assuming that the normal-flow (DFSYN) requests were not set to CS earlier). Therefore, switching of the CA/CS mode only applies to the RTYPE indicated and a separate CA/CS mode setting exists for each type of input (DFSYN, DFASY, and RESP).

OPTCD=SPEC | ANY
Indicates whether the RECEIVE macro instruction can only be satisfied by input from a specific logical unit (SPEC) or whether it can be satisfied by input from any connected logical unit that is in continue-any mode (ANY).

When OPTCD=SPEC is used, the logical unit's CID must be in the RPL when the macro instruction is executed. When OPTCD=ANY is specified, input from a logical unit in continue-any mode can satisfy a RECEIVE issued with RTYPE=DFASY or RTYPE=RESP only if PROC=NDFASYX or PROC=NRESPX (respectively) was specified in the NIB. See the descriptions of DFASY and RESP exit routines for more information.

A RECEIVE macro instruction with OPTCD=ANY can be issued before terminals are connected to the application program if the application program has opened an ACB. The receive request is queued until one or more terminals are connected and data arrives from any of the terminals in continue-any mode. If a receive request with OPTCD=ANY has not been completed and the application program issues CLSDST macro instructions disconnecting its terminals but does not close the ACB, the receive request does not have to be reissued after a subsequent OPNDST is issued.

At the completion of the RECEIVE macro instruction, the ARG field contains the CID of the logical unit whose input satisfied the RECEIVE.

OPTCD=TRUNC | KEEP | NIBTK
Indicates whether overlength input data is to be truncated (TRUNC), kept (KEEP), or whether the PROC=TRUNC | KEEP setting in the terminal's NIB is to be used to determine whether the input is to be truncated or kept.

Overlength input data is data whose length exceeds the value set in the AREALEN field of the RECEIVE macro instruction's RPL. When overlength data is truncated, the macro instruction is completed and the excess data is lost.

When overlength data is kept, the macro instruction is completed normally, and RECLEN is set to indicate the total amount unreceived of data. One or more additional RECEIVE macro instructions are required to obtain the excess data. After each RECEIVE, the value of RECLEN is decreased by the amount of data received. When AREALEN=0 is set, the entire input is kept.

OPTCD=Q | NQ
Indicates the action to be taken if no input (of the type specified by the RTYPE operand) is available when the macro instruction is executed. OPTCD=Q means that the macro instruction is to be completed when the appropriate input eventually arrives. OPTCD=NQ means that the macro instruction is to be completed immediately with RTNCD=0 and FDBK2=6 if the input is not available.

RTYPE=([DFSYN | NDFSYN] [,DFASY | NDFASY] [,RESP | NRESP])
Indicates the types of input that can satisfy this RECEIVE macro instruction. DFSYN means that data and other normal-flow messages can satisfy the RECEIVE macro. DFASY means that expedited-flow messages can satisfy the RECEIVE macro. RESP means that responses to data messages can satisfy the RECEIVE macro. The negative settings (NDFSYN, NDFASY, and NRESP) indicate that the corresponding type of input cannot satisfy the RECEIVE macro. For explanations of the normal-flow and expedited-flow messages, see *VTAM Concepts and Planning.*

Example

```
RCV1     RECEIVE  RPL=RPL1,AREA=INBUF,AREALEN=128,
                  RTYPE=(DFSYN,DFASY,NRESP),
                  OPTCD=(ANY,Q,NIBTK)
```

RCV1 is completed when an incoming message (normal-flow or expedited-flow) is available from any logical unit that is in CA mode for that RTYPE. Responses cannot cause RCV1 to be completed. After RCV1 is completed, the application program can examine the CONTROL field of RPL1 to determine the type of message received. If a data message is received (CONTROL=DATA), the data is placed in INBUF. The TRUNC-KEEP setting in the logical unit's NIB determines what will be done with any data that exceeds 128 bytes.

Return of Status Information

After the RECEIVE operation is completed, the following RPL fields may be set by VTAM:

If RECEIVE was issued with OPTCD=ANY, the ARG field contains the CID of the logical unit whose input causes the macro instruction to be completed. If RECEIVE was issued with OPTCD=SPEC, the ARG field still contains the CID that was placed there prior to the execution of the macro instruction.

The RTYPE field indicates the type of input that satisfied the RECEIVE macro instruction. Only one type can satisfy the RECEIVE, even though more than one type may be eligible to satisfy the RECEIVE. Other RPL fields may be set depending on the type of received input, as shown below:

	Value of RTYPE		
Field	**DFSYN**	**DFASY**	**RESP**
ARG	X	X	X
RECLEN	X	-	-
SEQNO	X	X	X
RESPOND	X	X	X
USER	X	X	X
REQ	X	X	X
RTNCD	X	X	X
FDBK2	X	X	X
CODESEL	X	-	-
CHNGDIR	X	X[1]	X[1]
BRACKET	X	X[1]	X[1]
CHAIN	X	-	-
SIGDATA	-	X	-
CONTROL	X	X	X
OPTCD	X	-	X
USENSEI	X[2]	-	X
SSENSEI	X[2]	-	X
SSENSMI	X[2]	-	X

[1] Although VTAM and certain logical units support this RTYPE, it is not supported by SNA.

[2] For exception requests and Logical Unit Status commands only.

The RECLEN field indicates the number of bytes of data received by VTAM. VTAM has moved as much of this data as possible into the input data area pointed to by the AREA field. If KEEP is in effect and the value in the RECLEN field exceeds the value in the AREALEN field, there is excess data present that can be obtained with more RECEIVE macro instructions. The value in RECLEN indicates the total number of bytes received by VTAM decremented by the total number of bytes moved by any previous RECEIVE macro instruction.

The SEQNO field contains the sequence number of the message or response.

The RESPOND field indicates either the type of response that is received (if RTYPE=RESP), or the type of response that the logical unit expects in reply if RTYPE=DFSYN), or it is a notification to the application program at a response that VTAM has already sent if RTYPE=DFSYN).

When a response is received, the RESPOND field for each RPL used to receive the data request (if more than one is used) indicates the following:

RESPOND=EX,FME,RRN	This is a negative response with response type 1 and 2 set.
RESPOND=EX,FME,NRRN	This is a negative response with response type 1 set.
RESPOND=EX,NFME,RRN	This is a negative response with response type 2 set.
RESPOND=EX,NFME,NRRN	Invalid.
RESPOND=NEX,FME,RRN	This is a positive response with response type 1 and 2 set.
RESPOND=NEX,FME,NRRN	This is a positive response with response type 1 set.
RESPOND=NEX,NFME,RRN	This is a positive response with response type 2 set.
RESPOND=NEX,NFME,NRRN	Invalid.

When a message is received, the RESPOND field indicates the type of response that is required (the value in the RTNCD field indicates whether the message was received successfully).

RESPOND=EX,FME,RRN	If the message is processed successfully, no response is sent; if the message is not processed successfully, return a negative response with response type 1 and 2 set.
RESPOND=EX,FME,NRRN	If the message is processed successfully, no response is sent; if the message is not processed successfully, return a negative response with response type 1 set.
RESPOND=EX,NFME,RRN	If the message is processed successfully, no response is sent; if the message is not processed successfully, return a negative response with response type 2 set.
RESPOND=EX,NFME,NRRN	Invalid.
RESPOND=NEX,FME,RRN	Return a positive or negative response, as appropriate, with response type 1 and 2 set.
RESPOND=NEX,FME,NRR	Return a positive or negative response, as appropriate, with response type 1 set.
RESPOND=NEX,NFME,RRN	Return a positive or negative response, as appropriate, with response type 2 set.
RESPOND=NEX,NFME,NRRN	Return no response.

See the *Macro Language Guide* for more information.

If VTAM receives a data request or data response that indicates an FM header is present, OPTCD=FMHDR is set.

The USER field contains the value that was originally set in the USERFLD field of the logical unit's NIB.

The value 35 (decimal) is set in the REQ field, indicating a RECEIVE request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C. Registers 0 and 15 are also set as indicated in Appendix C.

The CODESEL field indicates whether the input is in EBCDIC (STANDARD) or in some other code (ALT). ASCII is the only other type of code presently supported.

The CHNGDIR field indicates whether a Change Direction Request or a Change Direction Command indicator is present:

CHNGDIR=(CMD,NREQ)	A Change Direction Command indicator is present; the terminal was the sender and it has changed direction so that the application program can now transmit normal-flow requests (DFSYN only).
CHNGDIR=(NCMD,REQ)	A Change Direction Request indicator is present; the application program is currently sending and the terminal is requesting the application program to change direction by returning a Change Direction Command indicator and permit the terminal to send. (Although VTAM and certain logical units support this use of change direction, this does not correspond to the SNA requirements for this protocol.)
CHNGDIR=(CMD,REQ)	Invalid.
CHNGDIR=(NCMD,NREQ)	Neither indicator is present.

The BRACKET field indicates whether the current bracket is beginning, ending, or continuing (DFSYN only):

BRACKET=(BB,NEB)	The input is the first of a new bracket.
BRACKET=(NBB,NEB)	The input is a continuation of the current bracket. This indicator is also set when brackets are not being used.
BRACKET=(NBB,EB)	The input is the end of the current bracket.
BRACKET=(BB,EB)	The input itself constitutes an entire bracket.

The CHAIN field indicates the message's relative position within the chain being sent to the application program (DFSYN only):

CHAIN=FIRST	The message is the first of a new chain.
CHAIN=MIDDLE	The message is a continuation of the current chain.
CHAIN=LAST.	The message is the last of the current chain.
CHAIN=ONLY	The message itself constitutes an entire chain.

The SIGDATA field contains 4 bytes of signal information. This field is set when the RECEIVE is completed with CONTROL=SIGNAL.

The CONTROL field indicates the presence of data or data flow control commands in the message:

CONTROL=	Possible when RTYPE=	Meaning
DATA	DFSYN	A data message has been received.
QEC	DFASY	
RELQ	DFASY	
QC	DFSYN	
CANCEL	DFSYN	
CHASE	DFSYN	A data flow control command has been received.
LUS	DFSYN	
SIGNAL	DFASY	
RTR	DFSYN	
RSHUTD	DFASY	
SHUTC	DFASY	

When a negative response or Logical Unit Status (LUS) command has been received, the SSENSEI field may contain a system sense value or it is set to 0. See Appendix C for an explanation of the SSENSEI codes.

When the SSENSEI field is set, the SSENSMI field may also be set. The SSENSMI field contains a system sense modifier value; when combined with the SSENSEI value, a specific type of error is identified. The SSENSMI value is tested as a 1-byte binary value. See Appendix C for a list of the SSENSMI values.

When a negative response or Logical Unit Status (LUS) command has been received, the USENSEI field contains a 2-byte user sense value. This value is tested as a 2-byte binary value.

RESET–Cancel an I/O Operation (Basic Mode Only)

The RESET macro instruction can be used to:

- Cancel an I/O operation that is pending, but is not yet in the process of being completed (that is, no data transfer activity has yet begun). This form of RESET is selected by setting the COND option code.
- Cancel an I/O operation, whether it is pending or in the process of being completed, and in addition reset any error lock that may have been set for the terminal. This form of RESET is selected by setting the UNCOND option code.
- Reset any error lock that may have been set for the terminal, without canceling any pending I/O operation. This form of RESET is selected by setting the LOCK option code.

When a request is canceled, VTAM "completes" the canceled request (that is, returns control, posts the ECB, or schedules an RPL exit routine, as indicated by the SYN-ASY option code and the ECB-EXIT field) with a return code indicating that a RESET caused the request to be terminated. (The completion of the canceled request and the completion of RESET occur independently of each other; it is impossible at assembly time to know which will complete first.)

Name	*Operation*	*Operands*
[symbol]	**RESET**	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address

Indicates the location of the RPL that governs the execution of the RESET macro instruction.

rpl field name=new value

Indicates an RPL field to be modified and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the RESET macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. ARG can also be coded. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register. The value supplied for the ARG keyword must indicate a register.

The following RPL operands apply to a RESET macro instruction:

ACB=acb address

Indicates the ACB that identifies the application program.

ARG=(register)

Indicates the register containing the terminal's CID. The ARG field of RESET's RPL must contain the CID of the terminal whose I/O operation is to be canceled or whose error lock is to be reset. Register notation is used to place the CID into the ARG field with this RESET macro instruction. Note that you do not issue RESET for a particular request; you issue RESET for a specific *terminal*, and let VTAM deal with any requests that may be outstanding for that terminal.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) RESET macro instruction is completed. For OPTCD=LOCK, the macro instruction is completed when the error lock has been reset. For OPTCD=COND or OPTCD=UNCOND, the macro instruction is completed when all outstanding I/O requests to the terminal have been posted complete. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When the SYN option code is set, control is returned to the application program when the macro instruction has been completed. When ASY is set, control is returned as soon as VTAM has accepted the request. Once the macro instruction has been completed, the ECB is posted or the RPL exit-routine is scheduled, as indicated by the ECB-EXIT field.

OPTCD=CA | CS
When CA is set, data obtained from the terminal can satisfy a READ macro instruction. When CS is set, only READ (OPTCD=SPEC) macro instructions can obtain data from the terminal. See the RPL macro instruction for more information.

OPTCD=COND | UNCOND | LOCK

COND
RESET cancels any I/O operation that has been initiated, but for which no data has been transferred. If data transfer is in progress when RESET is executed, the RPL's RTNCD and FDBK2 fields are set to indicate that cancelation did not occur (RTNCD=0, FDBK2=1). If an I/O operation is pending, that operation is posted as completed (IO=COMPLETE if an internal ECB was used), and the RTNCD and FDBK2 fields of that request's RPL indicate that RESET caused the premature completion of the operation. The RESET RPL itself is posted to indicate normal completion. The RESET RPL is also posted to indicate normal completion if there is no I/O in progress to be canceled.

OPTCD=COND cannot be used if an error lock has been set for the terminal. Use one of the other forms of RESET to reset the error lock. (FDBK2 codes returned from the I/O request indicate whether the I/O operation failed and, if so, whether an error lock was set.) OPTCD=COND is appropriate when the application program wants to write to a terminal only if no data is being sent (and can tolerate a resulting delay).

The RESET operation is completed when all of the pending I/O operations for the terminal have been canceled (or is completed immediately if VTAM determines that I/O is in progress).

UNCOND

RESET cancels any I/O operation, pending or otherwise, that is being performed with the terminal. If an internal ECB was used, the RPL is set to IO=COMPLETE. (If there is no I/O operation to be canceled, RESET completes normally.) Any data that a canceled solicit operation has already brought into VTAM storage buffers is available for retrieval by the application program. Data that is being sent or is about to be sent, however, may be lost. When a solicit, read, or write operation is canceled, that operation is posted as completed, and the RTNCD and FDBK2 fields of its RPL indicate that RESET caused the premature completion of the operation. OPTCD=UNCOND also causes RESET to perform the same resetting operation indicated below with OPTCD=LOCK. OPTCD=UNCOND is appropriate when a terminal is being solicited for input, but the application program wants to immediately write to the terminal without delay (and can tolerate a possible loss of data).

OPTCD=UNCOND causes the communications controller to do the following: For start-stop devices with the break feature a reset immediate is sent, and for other start-stop devices, a reset ahead-of-command is sent; for BSC devices, a reset orderly (RVI) is sent. Any outstanding WRITE operations to the terminal are posted completed, with their return codes indicating that the operation was canceled by RESET. The OPTCD=UNCOND will not necessarily always perform the reset unconditionally when issued to a BSC device the first time. An unsuccessful return code may occur. When this occurs, reissue the RESET until it is successful.

The RESET operation is completed when all of the I/O requests for the terminal have been canceled. If a RESET macro instruction is issued to cancel an I/O operation, an EOT must be sent to release the line.

If a read request is pending for a binary synchronous device at the time RESET is issued, the application program must continue to issue read requests until an EOT is received. (The FDBK field is set upon receipt of an EOT.) If a read request is pending for a start-stop device without the break feature, the application program must continue to issue read requests until the amount of data solicited from the device has been obtained. That is, if PROC=TRANS is in effect for SOLICIT, reads must be issued until an EOT is received; if PROC=MSG is in effect, reads must be issued until EOM is received, and so forth. If a read request is pending for a start-stop device *with* the break feature, the application program must allow that read to complete before issuing RESET, but no further reads need be issued. (If that read results in excess data being received, a second read to obtain that excess data would have to be issued, however, before RESET could be issued.)

LOCK

RESET resets an error lock that has been set for the terminal. The RESET operation is completed as soon as the error lock is reset. Error locks are set by a communications controller when it determines that it should not or cannot continue to communicate with a terminal until the application program determines the next action to be performed.

If several WRITE requests have been issued and an error lock is set before all have been completed, resetting the error lock restarts the remaining WRITE operations.

Note: *This type of RESET should not be used if the error lock was set while a DO macro instruction involving more than one LDO was being executed. Use RESET with OPTCD=UNCOND instead.*

You can determine that an error lock has been set by examining the RTNCD and FDBK2 fields of each I/O request's RPL. The error lock is set if any of the following RTNCD-FDBK2 codes are returned (see Appendix C):

RTNCD	**FDBK2**	**Type of Error**
4	0	RVI received
4	1	Attention or reverse break received
12 (X'0C')	0	Error lock set
12 (X'0C')	1	Terminal not usable
12 (X'0C')	2	Request canceled by TRM
12 (X'0C')	6	NCP abended, restart successful
12 (X'0C')	15 (X'0F')	Yielded to contention
16 (X'10')	4	VTAM/NCP incompatibility
16 (X'10')	11 (X'0B')	Dial-out disconnection
16 (X'10')	12 (X'0C')	Dial-in disconnection
20 (X'14')	47 (X'2F')	Too many leading graphic characters
20 (X'14')	48 (X'30')	Invalid LEN field
20 (X'14')	49 (X'31')	Invalid data area
20 (X'14')	50 (X'32')	Request invalid for specified device
20 (X'14')	51 (X'33')	WRITE canceled (input arriving)
20 (X'14')	52 (X'34')	First I/O not READ or SOLICIT
20 (X'14')	53 (X'35')	Terminals not attached to same control unit
20 (X'14')	54 (X'36')	RESET (LOCK) invalid
20 (X'14')	55 (X'37')	Terminal not connected (copy LDO)
20 (X'14')	57 (X'39')	Invalid PROC option

Example

```
RESET1    RESET    RPL=RPL1,ARG=(3),ECB=ECBWORD,
                   OPTCD=(ASY,UNCOND)
           .
           .
           .
ECBWORD   DC       F(0)
RPL1      RPL      ACB=ACB1,AM=VTAM
```

RESET1 cancels any I/O operation pending or in progress for the terminal whose CID has been loaded into register 3. As soon as the cancellation has been scheduled, control is returned to the next instruction after RESET1. To verify that the cancellation has been completed, a CHECK macro instruction must be issued to determine if ECBWORD has been posted.

Return of Status Information

After the operation is completed, the following RPL fields are set:

The value 18 (decimal) is set in the REQ field, indicating a RESET request.

The USER field is set.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

RESETSR–Cancel RECEIVE Operations and Switch a Logical Unit's CA-CS Mode (Record Mode Only)

The RESETSR macro instruction is used to change the continue-any or continue-specific mode of a specified logical unit or BSC or local non-SNA 3270 terminal that is used in record mode and to cancel RECEIVE with OPTCD=SPEC macro instructions that are outstanding for the logical unit. Figure 10 summarizes the functions of RESETSR and their associated operands.

Changing CA-CS Mode

RESETSR changes a logical unit's continue-any (CA) or continue-specific (CS) mode in the same manner as do SEND and RECEIVE macro instructions.

When the CA-CS option code is set to CA, RESETSR places the logical unit into continue-any mode if it is not already in that mode. Continue-any mode means that RECEIVE macro instructions issued in the any-mode (OPTCD=ANY) as well as in the specific-mode (OPTCD=SPEC) can be satisfied by input from the logical unit.

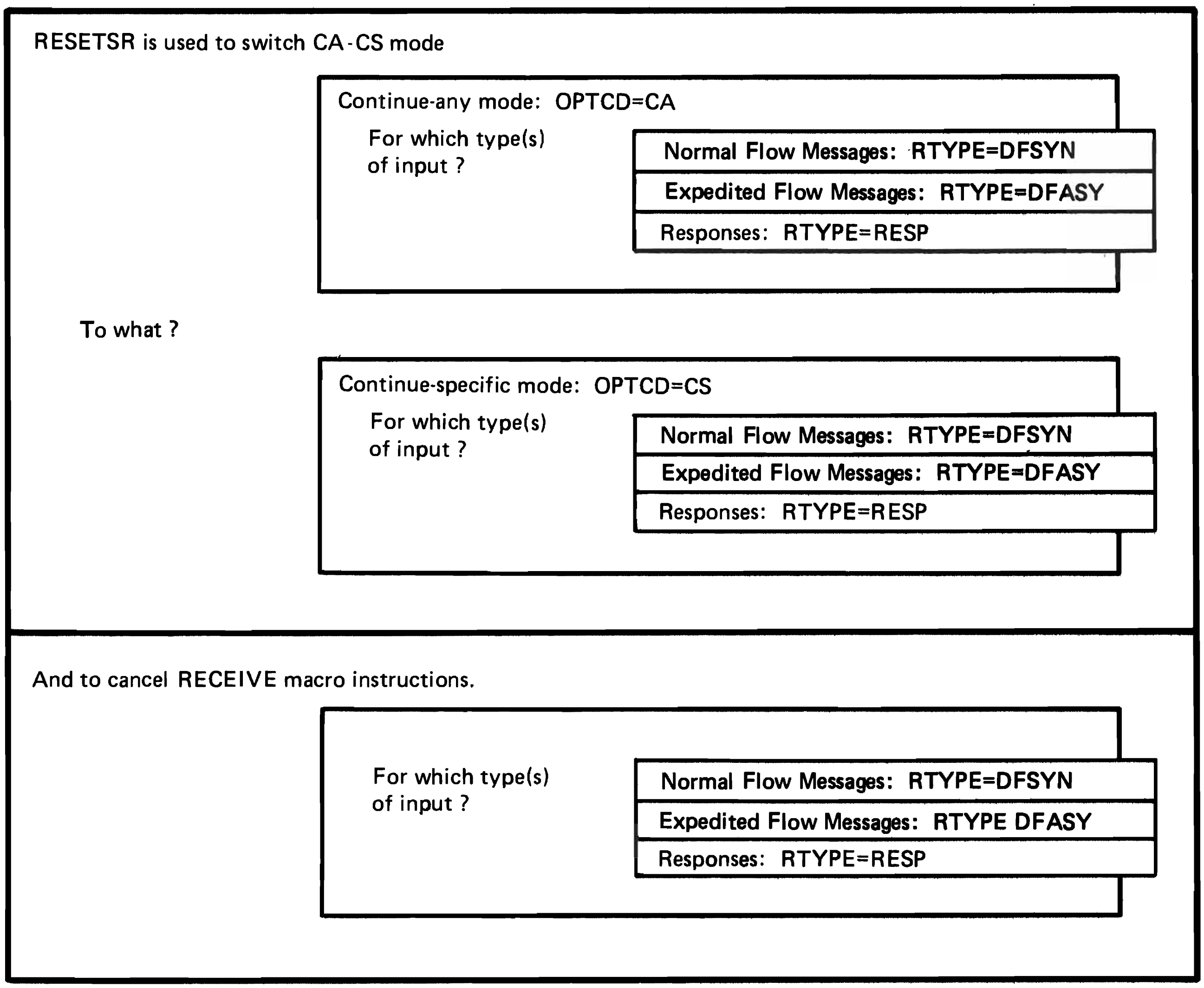


Figure 10. The Major RESETSR Options

When the CA-CS option code is set to CS, RESETSR places the logical unit into continue-specific mode, if it is not already in that mode. Continue-specific mode means that *only* RECEIVE macro instructions issued in the specific-mode can be satisfied by input from the logical unit.

A logical unit's CA-CS mode does not apply generally to all input from the logical unit, but applies individually for the three types of input from the logical unit—normal-flow messages, expedited-flow messages, and responses. The application program selects the type or types of input by setting the RPL's RTYPE field.

For example, suppose that RESETSR is issued with the CA-CS option set to CS (change to CS mode), and the RTYPE field set to DFASY (expedited-flow messages). When the RESETSR macro instruction is completed, the logical unit is placed in continue-specific mode for expedited-flow messages. This would mean that expedited-flow messages sent by the logical unit could not satisfy a RECEIVE issued in the any-mode; they could only satisfy a RECEIVE macro instruction issued in the specific-mode for expedited flow messages.

If a RESETSR, issued to change a logical unit's CA-CS mode, completes successfully (that is, actually changes the mode) and there is a pending SEND or RECEIVE, the SEND or RECEIVE may not complete successfully. The mode specified in the RESETSR may conflict with the OPTCD=ANY or OPTCD=SPEC that is specified in the pending SEND or RECEIVE macro instruction.

Note: *If program performance is a critical factor, it may be more efficient to change the CA-CS mode in the SEND macro for the last response.*

Canceling RECEIVE Requests

The RTYPE field of the RESETSR's RPL indicates, for the designated terminal, the type or types of RECEIVE requests that are canceled. For every RTYPE specified in the RESETSR macro, VTAM sets the corresponding RTYPE operand to its negative value (NDFSYN, NDFASY, NRESP) in each pending RECEIVE (with OPTCD=SPEC) for the terminal. A RECEIVE is canceled if the combination of input types specified in its RPL is included in those specified in the RESETSR macro instruction.

For example, suppose that these three specific RECEIVE macro instructions are pending:

```
RCV1   RECEIVE   RPL=RPL1,RTYPE=(DFSYN,NDFASY,NRESP)
RCV2   RECEIVE   RPL=RPL2,RTYPE=(DFSYN,DFASY,NRESP)
RCV3   RECEIVE   RPL=RPL3,RTYPE=(DFSYN,DFASY,RESP)
```

The following RESETSR macro instruction would change all DFSYN values to NDFSYN and all DFASY values to NDFASY:

```
RST    RESETSR   RPL=RPL4,RTYPE=(DFSYN,DFASY)
```

Since the three RECEIVE macros would in effect now be set as follows, RCV1 and RCV2 would be canceled (all three RTYPE operands are negative), but RCV3 would not be canceled:

```
RCV1   RECEIVE   RPL=RPL1,RTYPE=(NDFSYN,NDFASY,NRESP)
RCV2   RECEIVE   RPL=RPL2,RTYPE=(NDFSYN,NDFASY,NRESP)
RCV3   RECEIVE   RPL=RPL3,RTYPE=(NDFSYN,NDFASY,RESP)
```

When a RECEIVE is canceled, its RPL is posted complete (that is control is returned, its ECB is posted, or its exit routine is scheduled) with RTNCD=12 and FDBK2=10. The OPTCD=CA|CS setting of each canceled RPL is ignored.

Note: *If a CA-CS mode is also specifid, it may change the CA-CS mode as described above.*

Name	*Operation*	*Operands*
[symbol]	**RESETSR**	**RPL=rpl address** **[,rpl field name=new value]** . . .

RPL=rpl address

Indicates the location of the RPL that describes the RESETSR operation.

rpl field name=new value

Indicates an RPL field to be modified and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with this RESETSR macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. ARG can also be coded. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or register notation may be used.

The following RPL operands apply to the RESETSR macro instruction:

ACB=acb address

Indicates the ACB that identifies the application program and was used when the logical unit was connected.

ARG=register

The RESETSR macro instruction is always directed toward a specific logical unit. The ARG operand specifies the register containing the CID of that logical unit. If the ARG field is not modified, the CID already in the RPL's ARG field is used.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address

Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) RESETSR request is completed. A RESETSR request is completed when the appropriate macro instructions have been canceled, and the logical unit's CA-CS mode has been set. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted and CHECK or WAIT must be used to determine when posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

BRANCH=YES | NO

If RESETSR is to be issued in an application program that is running in privileged state under a TCB (OS/VS2 MVS only), BRANCH can be set to YES. See the RPL macro instruction for more information.

OPTCD=SYN | ASY

When SYN is set, control is returned to the application program when the RESETSR operations have been completed. When ASY is set, control is returned as soon as VTAM has accepted the RESETSR request; once the operations have been completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB-EXIT field. See the RPL macro instruction for more information.

OPTCD=CA | CS

This option code determines whether the logical unit is placed in continue-any (CA) or continue-specific (CS) mode. The new CA-CS mode applies to the type of input specfied in the RTYPE field. CA-CS mode is explained in the RPL macro instruction and in *VTAM Concepts and Planning.*

RTYPE=([DFSYN | NDFSYN] [,DFASY | NDFASY] [,RESP | NRESP])

The RTYPE operand indicates the type of input to be affected by the resetting of the logical unit's continue-any or continue-specific mode and which outstanding RECEIVE macros are to be canceled. (Continue-any mode means that input from the logical unit can satisfy a RECEIVE issued in the any-mode; continue-specific mode means that it cannot.)

DFSYN

The logical unit's CA-CS mode applies to normal-flow messages; NDFSYN means that the logical unit's CA-CS mode for normal-flow messages is not affected.

DFASY

The logical unit's CA-CS mode applies to expedited-flow messages; NDFASY means that the logical unit's CA-CS mode for expedited-flow messages is not affected.

RESP

The logical unit's CA-CS mode applies to response units; NRESP means that the logical unit's CA-CS mode for responses is not affected.

The RTYPE operand also designates the type of pending RECEIVE to be canceled. A RECEIVE request is canceled, however, only if all of the input types specified for the RECEIVE requests's RTYPE field are also included among those specified on the RESETSR request's RTYPE field. When the RECEIVE request is canceled, its RPL is posted complete with RTNCD=12 and FDBK2=10.

DFSYN

Pending RECEIVE requests for data messages or other normal-flow messages are canceled; NDFSYN means that RECEIVE requests for this type of input are not canceled.

DFASY

Pending RECEIVE requests for expedited-flow messages are canceled; NDFASY means that that RECEIVE requests for this type of input are not canceled.

RESP

Pending RECEIVE requests for responses are canceled; NRESP means that RECEIVE requests for responses are not canceled.

Example

```
RST1        RESETSR    RPL=RPL1,OPTCD=CA,
                       RTYPE=(DFSYN,NDFASY,NRESP)
```

RST1 cancels pending RECEIVE (OPTCD=SPEC, RTYPE=DFSYN) macro instructions for the logical unit identified in RPL1's ARG field. RST1 also switches the terminal's CA-CS mode for normal-flow messages to continue-any (CA) mode. This means that either a RECEIVE DFSYN with OPTCD=SPEC or with OPTCD=ANY can be issued to obtain normal-flow input from the logical unit. The logical unit's CA-CS mode for DFASY and RESP input is not affected; RECEIVE macro instructions for these request types are also not affected.

Return of Status Information

After the RESETSR operation is completed, the following RPL fields are set:

The value 36 (decimal) is set in the REQ field, indicating a RESETSR request.

The value originally set in the USERFLD field of the logical unit's NIB is set in the USER field of the RPL.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

RPL—Create a Request Parameter List

Every request that an application program makes for connection or for I/O operations must refer to an RPL. A request parameter list, or RPL, is a control block used by the application program to describe the requests it makes to VTAM. The application program may, for example, simply issue a RECEIVE macro and indicate an RPL; it is the RPL that shows VTAM which terminal the input is to be obtained from, where the input data is to be placed, how the application program is to be notified when the operation is completed, and a variety of other options to be followed while the request is being processed. If the RPL already contains a request code in its REQ field, an EXECRPL macro can be used in place of the RPL-based macro indicated in REQ.

An application program can create many RPLs; a separate RPL can, in fact, be created for every connection and I/O request in the application program. Or, at the other extreme, one RPL could serve for all connection and I/O requests in the program (assuming that all the requests were synchronous —that is, issued with OPTCD=SYN set). This multiple use of an RPL is possible because each connection and I/O request can itself modify fields of the RPL to which it points. The RPL can thus be thought of as the list form of all of the connection and I/O macros.

The RPL macro instruction builds an RPL during assembly. The RPL is built on a fullword boundary. An RPL can also be generated during program execution with the GENCB macro instruction. See GENCB for a description of this facility.

Requests for RPL modification can be made as part of a connection or I/O macro, or by the MODCB macro instruction. Either way involves naming an RPL field and specifying its new value. It is useful to keep in mind that every operand of the RPL macro represents a field in the RPL it generates. Subsequent requests to modify any RPL field use the keyword of the operand corresponding to the field being modified.

Assumed (default) values for most of the RPL fields are set by VTAM when the RPL is initially assembled or generated. These assumed values are noted in the operand descriptions below. Once an RPL field has been set, however, the field is never reset by VTAM to its original value (three exceptions to this rule—the SSENSEO, SSENSMO, and USENSEO fields—are noted below).

Although all of the RPL operands are optional (with the exception of AM=VTAM) and may be specified with any of the RPL-based macro instructions, all of the RPL-based macro instructions require that certain RPL fields be set when the macro instruction is executed. These fields are identified in Figure 10 at the end of this macro instruction description.

<table>
<tr><th>Name</th><th>Operation</th><th>Operands</th></tr>
<tr><td>[symbol]</td><td>RPL</td><td>AM=VTAM
[, ACB=acb address]
[, NIB=nib address]
[, AREA=data area address]
[, AREALEN=data area length]
[, RECLEN=data length]
[, AAREA=alternate data area address]
[, AAREALN=alternate data area length]
[{, ECB=event control block address
  , EXIT=rpl exit-routine address}]
[, BRANCH=YES[1] | NO]
[, SEQNO=sequence number]
[, POST=SCHED | RESP]
[, RESPOND=([EX | NEX] [,FME | NFME] [,RRN | NRRN])]
[, CONTROL=(DATA | QEC | RELQ | QC | CANCEL |
    CHASE | SHUTD | BID | LUS | SDT |
    CLEAR | STSN | SIGNAL)]
[, CHAIN=FIRST | MIDDLE | LAST | ONLY]
[, CHNGDIR=(CMD | NCMD, REQ | NREQ)]
[, BRACKET=(BB | NBB, EB | NEB)]
[, RTYPE=([DFSYN | NDFSYN] [,DFASY | NDFASY]
    [,RESP | NRESP])]
[, STYPE=REQ | RESP]
[, SSENSEO=0 | CPM | STATE | FI | RR][2]
[, SSENSMO=system sense modifier value][2]
[, USENSEO=user sense value][2]
[, IBSQAC=SET | TESTSET | INVALID | IGNORE]
[, OBSQAC=SET | TESTSET | INVALID | IGNORE]
[, IBSQVAL=inbound sequence number]
[, OBSQVAL=outbound sequence number]
[, SIGDATA=signal data]
[, CODESEL=STANDARD | ALT]
[, OPTCD=(
    [, NIBTK | TRUNC | KEEP]
    [, NFMHDR | FMHDR]
    [, CONALL | CONANY]
    [, ACCEPT | ACQUIRE]
    [, SPEC | ANY]
    [, QUIESCE | STOP | START]
    [, RELEASE | PASS]
    [, LOGONMSG | DEVCHAR |
      [,COUNTS | TERMS | APPSTAT |
      CIDXLATE | TOPLOGON |
      BSCID | SESSPARM]
    [, SYN | ASY]
    [, CA | CS]
    [, BLK | LBM | LBT[3]]
    [, NCONV | CONV]
    [, COND | UNCOND | LOCK]
    [, NERASE | ERASE | EAU]
    [, RELRQ | NRELRQ]
    [, Q | NQ])</td></tr>
</table>

1 The BRANCH=YES operand is valid only in OS/VS2 MVS.

2 These fields are cleared by VTAM each time the RPL is reset to its inactive state.

3 Since OPTCD=BLK is invalid for 3270 terminals, the default for those devices is OPTCD=LBT.

AM=VTAM

Indicates that a VTAM RPL is to be built. This operand is required.

ACB=acb address

Associates the request that will use this RPL with an ACB.

Format: Expressions involving registers cannot be used with the RPL macro instruction.

If you omit this operand, the ACB field is set to 0.

NIB=nib address

Identifies the NIB whose NAME field indicates the terminal that is to be the object of an OPNDST, CLSDST, INQUIRE, INTRPRET, CHANGE, or SIMLOGON macro instruction.

Although these macro instructions use a NIB address to indicate a terminal, the READ, RECEIVE (OPTCD=SPEC), SEND, RESETSR, SESSIONC, WRITE, SOLICIT, RESET, and DO macro instructions use a CID to indicate a terminal (and INTERPRET and CLSDST, along with some forms of INQUIRE, work either way). CIDs (communication identifiers) are supplied to the application program upon completion of an OPNDST macro instruction. The CID and the NIB address occupy the same physical field in the control block. VTAM can distinguish between a NIB address and a CID only through a particular bit set in the field. For this reason, the field is called the NIB field when a NIB address is being inserted into it, and an ARG field when a CID is being inserted into it. When NIB=*address* appears on a CHANGE macro instruction, for example, the bit is set to indicate that the field contains a NIB address. When ARG=(*register*) is coded on a READ macro instruction, for example, the bit is set to indicate that the field contains a CID. (Note that register notation must be used with ARG, since CIDs are not available until program execution.)

The point to remember when dealing with the NIB-ARG field is this: Since only one physical field is involved, always use the NIB keyword to insert a NIB address and always use the ARG keyword to insert a CID. This will set the RPL to indicate whether a NIB address or CID is present. This rule also applies to the GENCB and MODCB macro instructions.

If the NIB operand is coded in an RPL macro for a DO, RESET, SOLICIT, READ, RECEIVE, RESETSR, SEND, SESSIONC, or WRITE request, the request will be completed with an error code.

Format: Expressions involving registers cannot be used with the RPL macro instruction.

If you omit this operand, the NIB field is set to 0.

AREA=data area address

When used by a SIMLOGON, INTRPRET, or a CLSDST with OPTCD= PASS macro instruction, AREA indicates the address of an area containing a logon message.

When used by a SEND, RECEIVE, READ, or WRITE macro instruction, AREA indicates the address of an area in program storage into which data is to be read or from which data is to be written.

When used by an INQUIRE macro instruction, AREA indicates where the data obtained by INQUIRE is to be placed.

When used by a DO macro instruction, AREA contains the address of an LDO.

When used by a RCVCMD macro instruction, AREA contains the address of an area into which a header and a VTAM network operator message will be placed.

When used by a SENDCMD macro instruction, AREA contains the address of an area containing a header and a VTAM network operator command.

The AREA field is also set upon return from an OPNDST (OPTCD=ACQUIRE) macro instruction, indicating the address of a NIB or list of NIBs. The AREA field is not set by the application program before OPNDST is issued.

Format: Expressions involving registers cannot be used with the RPL macro instruction.

If you omit this operand, the AREA field is set to 0.

AREALEN=data area length

Indicates the length (in bytes) of the data area identified by the AREA operand. The AREALEN operand is meaningful only for input operations or for the INQUIRE macro instruction; VTAM uses this length to determine whether the data it is placing in the area is too long to fit. For the RECEIVE macro instruction, AREALEN=0 means that no input data area is available.

Format: Expressions involving registers cannot be used with the RPL macro instruction.

If you omit this operand, the AREALEN field is set to 0.

RECLEN=data length

When used by a SIMLOGON or INTRPRET macro instruction, or by a CLSDST macro with the PASS option code, RECLEN indicates the length (in bytes) of a logon message or sequence contained in the area indicated by the AREA operand.

When used by a RECEIVE, SEND, SENDCMD, READ, or WRITE macro instruction, RECLEN indicates the length (in bytes) of the data that begins at the address indicated by AREA. For SEND, SENDCMD, and WRITE operations, RECLEN provides the application program a means of telling VTAM how much data is to be transferred. Users of SEND should take a particular care to insure that the RECLEN field is not improperly set when the macro is issued. The possible consequence of an excessive RECLEN value is described in the SEND macro instruction under RECLEN.

For RECEIVE and READ operations, the RECLEN *operand* has no meaning; but the 4-byte *field* in the RPL corresponding to RECLEN is set by VTAM when the input operation is finished to indicate the length of data that VTAM has just placed into AREA (for READ) or the total length of available data (for RECEIVE). For a conversational WRITE, which includes both an input and an output operation, RECLEN indicates the amount of data to be written. VTAM will post the length of the incoming data in an RPL field called the ARECLEN field.

When a RECEIVE operation is completed and excess data is available (that is, KEEP is in effect and the message is too long to fit in the input area), RECLEN contains the total length of the message. The application program can reissue the RECEIVE until the value in RECLEN is less than or equal to the value in AREALEN.

The RECLEN field is also set upon return from the SETLOGON macro instruction, indicating the number of logon requests currently queued for the application program. The RECLEN field is not set by the application program before SETLOGON is issued.

When a DO macro instruction for a READ LDO or a READBUF LDO is completed, the RECLEN field indicates the amount of data obtained from the terminal and placed in the data area in the LDO.

When an INQUIRE macro instruction is completed, the RECLEN field indicates the amount of data placed into AREA. If the amount of data is larger than the AREA field, RECLEN indicates the total length of the data. The INQUIRE can be reissued with the correct length specified.

The application program can obtain the value in the RECLEN field by issuing a SHOWCB macro, or it can test the contents of RECLEN against a fixed value with the TESTCB macro instruction. For example:

```
SHOWCB    RPL=(1),AM=VTAM      OBTAIN THIS RPL'S...
          FIELDS=RECLEN,       ...RECLEN FIELD...
          AREA=WORKAREA,       ...AND PLACE IT IN WORKAREA...
          LENGTH=4             ...WHICH IS FOUR BYTES LONG.
```

Format: Expressions involving registers cannot be used with the RPL macro instruction.

If you omit this operand, the RECLEN field is set to 0.

AAREA=alternate data area address

When used by a CLSDST macro instruction with a PASS option code, AAREA indicates the location of an 8-byte area containing the symbolic name of the application program to which a logon request is to be directed. The EBCDIC name should be left-justified and padded to the right with blanks. This name is the same as the name of the application program's APPL entry in the resource definition table.

When used by an INTRPRET macro instruction, AAREA indicates a work area where VTAM places the interpreted data sequence. See the INTRPRET macro instruction for details.

When used by a WRITE macro instruction with a CONV option code, AAREA indicates an input area in the application program into which data is to be placed. This type of operation is called a conversational write operation and is described in the WRITE macro instruction description.

When used by the DO macro instruction, AAREA indicates the address of the last LDO used.

Format: Expressions involving registers cannot be used with the RPL macro instruction.

If you omit this operand, the AAREA field is set to 0.

AAREALN=alternate data area length

Indicates the length (in bytes) of the data area identified by the AAREA operand. When AAREA is used as an input area for a INTRPRET or conversational WRITE macro instruction, VTAM uses this length to determine whether the data to be placed there is too long to fit.

Format: Expressions involving registers cannot be used with the RPL macro instruction.

If you omit this operand, the AAREALN field is set to 0.

ECB=event control block address

Indicates the location of an event control block (ECB) to be posted by VTAM when the connection or I/O request associated with this RPL is completed. The ECB can be any fullword of storage aligned on a fullword boundary.

Format: Expressions involving registers cannot be used with the RPL macro instruction.

The ECB field and the EXIT field share the same RPL field. If *asynchronous* handling of the connection or I/O request has been specified (ASY option code in the RPL), the ECB-EXIT field is used in this manner:

- If you specify ECB=*address*, VTAM uses the field as the address of an external ECB; you check and clear this ECB yourself (with CHECK, for example).
- If you specify EXIT=*address*, VTAM uses the field as the address of the RPL exit routine, and schedules the routine as indicated below (under EXIT operand).
- If you specify neither ECB=*address* nor EXIT=*address* VTAM uses the ECB-EXIT field as an internal ECB; you must issue CHECK for the RPL to check this ECB.

If *synchronous* handling has been specified (SYN option code in the RPL), the ECB-EXIT field is used in this manner:

- If you specify ECB=*address*, VTAM uses the field as the address of an external ECB; VTAM checks and clears this ECB itself.
- If you specify EXIT=*address*, VTAM uses the field as an internal ECB, thus destroying the exit routine address; VTAM checks and clears this ECB itself.
- If you specify neither ECB=*address* nor EXIT=*address* (this is the normal procedure for synchronous request handling), VTAM uses the ECB-EXIT field as an internal ECB; VTAM checks and clears this ECB itself.

VTAM clears *internal* ECBs (1) when it begins processing any RPL-based macro and (2) when the RPL is checked. However, VTAM clears *external* ECBs only when the RPL is checked. (RPL checking is done at request completion by VTAM for synchronous request handling, and is done by the user issuing CHECK for asynchronous request handling.) Users of external ECBs must therefore be sure that the external ECB is cleared (with CHECK or with assembler instructions) before the next RPL-based macro is issued.

EXIT=rpl exit routine address

Indicates the address of a routine to be scheduled when the request represented by this RPL is completed.

If the SYN option code has been specified, the exit routine is not used; should you specify an address anyway, the address is overwritten before the synchronous request completes. (VTAM uses the ECB-EXIT field as an internal ECB in this situation–see the ECB operand description above). The RPL exit routine is scheduled only if asychronous handling of the request has been specified.

When the routine receives control, it is passed the address of the RPL in register 1. The RTNCD and FDBK2 fields will indicate the status of the request.

The RTNCD-FDBK2 examination could reveal that the request was completed with a logical or physical error. You should issue CHECK in the RPL exit routine; this will schedule the LERAD or SYNAD exit routines, if appropriate, as well as set the RPL to an inactive state. (LERAD and SYNAD exits are discussed in the EXLST macro instruction description.) Never issue the CHECK in your main program unless you are sure that CHECK will be executed *after* the RPL exit routine is scheduled.

When the RPL exit routine receives control, these general purpose registers contain the following (registers 0 and 2-13 are unpredictable):

Register 1: The address of the RPL associated with the request whose completion has caused the RPL exit routine to be entered.

Register 14: The address in VTAM to which the RPL exit routine must branch when it is through processing. (For programs running under OS/VS2 in a privileged state, the address is an address in the OS/VS2 dispatcher, not in VTAM.)

Register 15: The address of the RPL exit routine.

No register save area is provided upon invocation of the RPL exit routine.

If the EXIT operand is specified, the ECB operand must not be specified. (The EXIT field and the ECB field occupy the same field in the RPL.)

BRANCH=YES|NO (OS/VS2 MVS only)

For OS/VS2 MVS application programs running in privileged state under a TCB, BRANCH indicates the type of processing to be used when a SEND, RECEIVE, RESETSR, or SESSIONC macro instruction is issued.

YES

When the macro instruction is issued, VTAM processes the macro instruction in an optimized high-priority manner. For OS/VS2 MVS programs running in privileged state under an SRB, rather than under a TCB, the macros are processed in this manner automatically regardless of the actual setting of the BRANCH field.

NO

When the macro instruction is issued, VTAM does not process the macro instruction in an optimized high-priority manner. For DOS/VS, OS/VS1, and OS/VS2 SVS programs, all requests are handled as though BRANCH=NO had been specified, regardless of the actual setting of the BRANCH field.

SEQNO=sequence number (Record mode only)

Indicates the sequence number of a response. When the application program sends a response to a terminal, the sequence number of the message being responded to is placed in the SEQNO field. This field is also set by VTAM on completion of a SEND (STYPE=REQ) and on completion of a RECEIVE.

Format: Specify any decimal value that does not exceed 65535 or specify a register (only the rightmost 2 bytes are used).

POST=SCHED | RESP (Record mode only)

This field is set when the application program sends a data message to a terminal and requests a definite response. It is ignored for STYPE=RESP or if no response or only an exception response is requested of STYPE=REQ. When POST=SCHED is used (scheduled output) the SEND operation is completed as soon as the output data area is free. The application program must issue a RECEIVE to obtain the response to the message. When POST=RESP is used (responded output) the SEND operation is not completed until a response to the message is returned. The response information is posted in the RPL fields of the SEND RPL.

RESPOND=([EX | NEX] [,FME | NFME] [,RRN | NRRN]) (Record mode only)

When a response is sent, the RESPOND field indicates the type of response–positive (NEX) or negative (EX)–and, whether it is response type 1 (FME) or response type 2 (RRN) or both.

When a message is sent, the RESPOND field indicates the *expected* response–definite (NEX) or exception only (EX)–and the type of the expected response–response type 1 (FME) or response type 2 (RRN) or both (FME,RRN).

CONTROL=(DATA | QEC | RELQ | QC | CANCEL | CHASE | SHUTD | BID | LUS | SDT | CLEAR | STSN | SIGNAL) (Record mode only)

Indicates whether data messages, data flow control commands, or session control commands are to be sent to a terminal. Data messages (DATA) and data flow control commands (QEC, RELQ, QC, CANCEL, CHASE, SHUTD, BID, LUS, and SIGNAL) are sent with the SEND macro instruction. The session control commands (SDT, CLEAR, and STSN) are issued by the SESSIONC macro instruction. See Chapter 5 of *VTAM Concepts and Planning* for an explanation of the commands designated by CONTROL.

CHAIN=FIRST | MIDDLE | LAST | ONLY (Record mode only)

This field is set when a message is sent to a terminal. It denotes the message's relative position within the chain currently being sent. ONLY means that the message is the sole element of the chain.

CHNGDIR=(CMD | NCMD ,REQ | NREQ) (Record mode only)

This field is set when a message or response is sent to a terminal. When CMD is set, a Change Direction Command indicator is included in the message or response. When REQ is set, a Change Direction Request indicator is included (although VTAM and certain logical units support this use of change direction. This does not correspond to the SNA requirements for this protocol).

BRACKET=(BB | NBB ,EB | NEB) (Record mode only)

This field is set when a message is sent to a terminal. When BB is set, a Begin Bracket indicator is included in the message. When EB is set, an End Bracket indicator is included. Note that both indicators can be included in one message.

If a bracket indicator is to be sent in a chain of messages, the indicator must be sent in the first message of that chain. For more information, see the description of these indicators in the sections describing the SEND and RECEIVE macros.

RTYPE=([DFSYN|NDFSYN] [,DFASY|NDFASY] [,RESP|NRESP]) (Record mode only)

When a RECEIVE macro instruction is issued, the RTYPE field designates the type or types of input eligible to satisfy the macro instruction (only one type can actually satisfy the RECEIVE). When a SEND or RESETSR macro instruction is issued, the RTYPE field indicates the type or types of input for which the terminal's CA-CS mode is to be switched.

DFSYN
Designates normal-flow messages. These include data messages and the Quiesce Complete, Cancel, Chase, Bid, Logical Unit Status, and Ready to Receive commands.

DFASY
Designates expedited-flow messages. These include the Quiesce at End of Chain, Release Quiesce, Shutdown, Requt Shutdown, Shutdown Complete, and Signal commands.

RESP
Designates normal-flow reponses.

Note: *Expedited-flow repsonses canbe received by the application program. They are intercepted by VTAM to complete SEND macro instructions that send expedited requests.*

STYPE=REQ|RESP (Record mode ly)
This field designates the typ outp be sent to a terminal. The application program uses STYPE=REQ equest nessage be sent. STYPE=RESP is used when a response is to be s STYPE must be set when a SESSIONC macro instruction is issued.

SSENSEO=0|CPM|STATE|FI Record mly)
This field is set when a respons gical Unit Status command is sent to a logical unit. Its pur tell the l unit the type of error that caused the exception condition. br types a ribed in Appendix C.

CPM
Designates a r error condi

STATE
Designates ndition.

FI
Design r condition.

RR
De ct condition.

the SSENSEO field is set to 0.

If t

Note: *When an RPL is assembled or generated, and each time the RPL is reset to its inactive state (that is, after each synchronous request with OPTCD=SYN or CHECK macro instruction), the SSENSEO field is cleared.*

SSENSMO=system sense modifier value (Record mode only)

This field is set when a negative response or a Logical Unit Status command is sent to a logical unit. The value set in this field is used in conjunction with the SSENSEO setting to describe the specific type of error that caused the exception condition. The meanings assigned to the SSENSMO values are described in Appendix C. If this operand is omitted, the SSENSMO field is set to 0.

Format: Specify any decimal value defined in Appendix C, specify a register (only the rightmost byte is used), or specify a 1-byte hexadecimal constant.

Examples: SSENSMO=1
SSENSMO=(7)
SSENSMO=X'1B'

Note: *When an RPL is assembled or generated, and each time the RPL is reset to its intive state (that is, after each synchronous request with OPTCD=SYN or CHEC macro instruction), the SSENSMO field is set to 0.*

USENSEO=r sense value (Record mode only)

This fi may beset when a negative response or Logical Unit Status command is sent t logical unit. The user sense field is user-defined and may be used to inform the lo unit that an exception condition is being raised by an application-programatedrror that is not a SNA defined error or it can be used to further modify SI defined system sense and system sense modifier values. See Appen fo more information. If this operand is omitted, the USENSEO field is set to

Form any decimal value that does not exceed 65535, specify a register (only ht 2 bytes are used), or specify a 2-byte hexadecimal or character const

Exam UO=13
U=(7)
U=X'4F4F'
U=C'ZZ'

Not *en assembled or generated, and each time the RPL is reset to its ive s, after each synchronous request with OPTCD=SYN or CH mac), the USENSEO field is set to 0.*

IBSQAC=T | T LID | IGNORE (Record mode only)
OBSQA ET | T LID | IGNORE (Record mode only)

The fields SSIONC macro instruction to designate which type of S and Test r command is being sent to a terminal. The setting of the IBSQAC the inbound sequence number; the **OBSQAC** field elates to the response number. See the SESSIONC macro instruction for d for each of the following:

SET

The seque terminal is made aware of the number, but possible re

TESTSET
The sequence number is reset. The logical unit is made aware of the number and returns a response regarding the validity of that number.

INVALID
The sequence number is not reset (the application program has lost its version of the sequence number). The terminal returns the sequence number.

IGNORE
The sequence number is not reset. No response is possible.

IBSQVAL=inbound sequence number (Record mode only)
OBSQVAL=outbound sequence number (Record mode only)
When SESSIONC is used to send an STSN command and SET or TESTSET is set in the IBSQAC or OBSQAC field, these fields contain the sequence number being reset or transmitted.

Format: Specify any decimal value that does not exceed 65535, any hexadecimal value that does not exceed FFFF, or specify a register (only the rightmost 2 bytes are used).

If this operand is omitted, this field is set to 0.

SIGDATA=signal data (Record mode only)
When the SEND macro is used to send the SIGNAL command to a logical unit, this field contains the signal data to be sent.

Format: Specify a decimal, hexadecimal, or character constant of from 1 to 4 bytes, or specify a register (the value in the register is used).

If this operand is omitted, this field is set to 0.

CODESEL=STANDARD|ALT (Record mode only)
This field indicates which data code is to be used in the request associated with this RPL. The application program and logical unit must have previously determined which type of code will be recognized as the standard code (such as EBCDIC) and which code will be recognized as the alternate code (such as ASCII).

STANDARD
Indicates the standard code is to be used

ALT
... the alternate code is to be used

... and I/O requests made usin...

If a bracket indicator is to be sent in a chain of messages, the indicator must be sent in the first message of that chain. For more information, see the description of these indicators in the sections describing the SEND and RECEIVE macros.

RTYPE=([DFSYN | NDFSYN] [,DFASY | NDFASY] [,RESP | NRESP])
(Record mode only)
When a RECEIVE macro instruction is issued, the RTYPE field designates the type or types of input eligible to satisfy the macro instruction (only one type can actually satisfy the RECEIVE). When a SEND or RESETSR macro instruction is issued, the RTYPE field indicates the type or types of input for which the terminal's CA-CS mode is to be switched.

DFSYN
Designates normal-flow messages. These include data messages and the Quiesce Complete, Cancel, Chase, Bid, Logical Unit Status, and Ready to Receive commands.

DFASY
Designates expedited-flow messages. These include the Quiesce at End of Chain, Release Quiesce, Shutdown, Request Shutdown, Shutdown Complete, and Signal commands.

RESP
Designates normal-flow responses.

Note: *Expedited-flow responses cannot be received by the application program. They are intercepted by VTAM to post complete SEND macro instructions that send expedited requests.*

STYPE=REQ | RESP (Record mode only)
This field designates the type of output to be sent to a terminal. The application program uses STYPE=REQ to request that a message be sent. STYPE=RESP is used when a response is to be sent. STYPE=REQ must be set when a SESSIONC macro instruction is issued.

SSENSEO=0 | CPM | STATE | FI | RR (Record mode only)
This field is set when a negative response or Logical Unit Status command is sent to a logical unit. Its purpose is to tell the logical unit the type of error that caused the exception condition. These error types are described in Appendix C.

CPM
Designates a request header error condition.

STATE
Designates a state error condition.

FI
Designates a request error condition.

RR
Designates a request reject condition.

If this operand is omitted, the SSENSEO field is set to 0.

Note: *When an RPL is assembled or generated, and each time the RPL is reset to its inactive state (that is, after each synchronous request with OPTCD=SYN or CHECK macro instruction), the SSENSEO field is cleared.*

SSENSMO=system sense modifier value (Record mode only)

This field is set when a negative response or a Logical Unit Status command is sent to a logical unit. The value set in this field is used in conjunction with the SSENSEO setting to describe the specific type of error that caused the exception condition. The meanings assigned to the SSENSMO values are described in Appendix C. If this operand is omitted, the SSENSMO field is set to 0.

Format: Specify any decimal value defined in Appendix C, specify a register (only the rightmost byte is used), or specify a 1-byte hexadecimal constant.

Examples: SSENSMO=1
SSENSMO=(7)
SSENSMO=X'1B'

Note: *When an RPL is assembled or generated, and each time the RPL is reset to its inactive state (that is, after each synchronous request with OPTCD=SYN or CHECK macro instruction), the SSENSMO field is set to 0.*

USENSEO=user sense value (Record mode only)

This field may be set when a negative response or Logical Unit Status command is sent to a logical unit. The user sense field is user-defined and may be used to inform the logical unit that an exception condition is being raised by an application-program-related error that is not a SNA defined error or it can be used to further modify the SNA defined system sense and system sense modifier values. See Appendix C for more information. If this operand is omitted, the USENSEO field is set to 0.

Format: Specify any decimal value that does not exceed 65535, specify a register (only the rightmost 2 bytes are used), or specify a 2-byte hexadecimal or character constant.

Examples: USENSEO=13
USENSEO=(7)
USENSEO=X'4F4F'
USENSEO=C'ZZ'

Note: *When the RPL is assembled or generated, and each time the RPL is reset to its inactive state (that is, after each synchronous request with OPTCD=SYN or CHECK macro instruction), the USENSEO field is set to 0.*

IBSQAC=<u>SET</u> | TESTSET | INVALID | IGNORE (Record mode only)
OBSQAC=<u>SET</u> | TESTSET | INVALID | IGNORE (Record mode only)

These fields are used by a SESSIONC macro instruction to designate which type of Set and Test Sequence Number command is being sent to a terminal. The setting of the IBSQAC field relates to the inbound sequence number; the OBSQAC field relates to the outbound sequence number. See the SESSIONC macro instruction for the responses that can be returned for each of the following:

SET

The sequence number is reset. The terminal is made aware of the number, but possible responses are limited.

TESTSET
The sequence number is reset. The logical unit is made aware of the number and returns a response regarding the validity of that number.

INVALID
The sequence number is not reset (the application program has lost its version of the sequence number). The terminal returns the sequence number.

IGNORE
The sequence number is not reset. No response is possible.

IBSQVAL=inbound sequence number (Record mode only)
OBSQVAL=outbound sequence number (Record mode only)
When SESSIONC is used to send an STSN command and SET or TESTSET is set in the IBSQAC or OBSQAC field, these fields contain the sequence number being reset or transmitted.

Format: Specify any decimal value that does not exceed 65535, any hexadecimal value that does not exceed FFFF, or specify a register (only the rightmost 2 bytes are used).

If this operand is omitted, this field is set to 0.

SIGDATA=signal data (Record mode only)
When the SEND macro is used to send the SIGNAL command to a logical unit, this field contains the signal data to be sent.

Format: Specify a decimal, hexadecimal, or character constant of from 1 to 4 bytes, or specify a register (the value in the register is used).

If this operand is omitted, this field is set to 0.

CODESEL=<u>STANDARD</u>|ALT (Record mode only)
This field indicates which data code is to be used in the request associated with this RPL. The application program and logical unit must have previously determined which type of code will be recognized as the standard code (such as EBCDIC) and which code will be recognized as the alternate code (such as ASCII).

STANDARD
Indicates the standard code is to be used

ALT
Indicates the alternate code is to be used

OPTCD=option code|(option code,. . .)
Indicates options that are to affect the connection and I/O requests made using this RPL.

Format: Code as indicated in the assembler format table. If only one option code is specified, the parentheses can be omitted.

```
RPL    ACB=ACB1,OPTCD=(SPEC,SYN,CS),AM=VTAM
RPL    ACB=ACB1,OPTCD=SPEC,AM=VTAM
```

Note: *The MODCB macro instruction can be used to change the option codes set in the RPL after it has been built.*

NIBTK | **TRUNC | KEEP** (Record mode only)
Indicates the action to be taken when a RECEIVE macro instruction is completed with input that is too large to fit in the input data area. TRUNC causes the excess data to be discarded. The application program is not notified that truncation occurred. KEEP causes the excess data to be saved for subsequent RECEIVE macro instructions. The application program can compare the value set in the RPL's RECLEN field (the amount of incoming data) with the value in the AREALEN field. If the RECLEN field is larger, excess data is present. NIBTK allows the TRUNC-KEEP processing option (see the NIB macro instruction) to determine whether excess data is to be kept or discarded.

NFMHDR | **FMHDR** (Record Mode only)
This option code indicates to VTAM how the format bit in the request header (RH) of this data message is to be set. This option applies only to data requests and data responses should be used to notify the terminal that the message contains or does not contain (FMHDR and NFMHDR, respectively) a user-defined function management header. If FMHDR is set, the format bit is set on in the request header and is delivered to the receiver.

CONALL | CONANY
When an OPNDST macro instruction (with an ACQUIRE option) is issued and the NIB field of its RPL indicates a list of NIBs, this option code indicates the following:

CONALL
Connection is to be made to *all* the available terminals in the list. The connections are made immediately.

CONANY
Connection is to be made to the first available terminal (if any) of the NIB list indicated by the NIB field. The request is completed when one connection has been made.

When a SIMLOGON macro instruction is issued and the NIB field of its RPL contains the address of a list of NIBs, this option code indicates the following:

CONALL
Logons are to be generated for *all* the terminals represented in the NIB list. The SIMLOGON operation is completed immediately. If Q is set, logons are generated as each terminal becomes available. If NQ is set and all the terminals are available, the logons are generated immediately; if all are not available, however, *no* logons are generated.

CONANY
A simulated logon is to be generated for the first available terminal of the NIB list. Control is passed to the application program's LOGON exit routine, if one exists, when this one logon has been generated. The parameter list passed to the LOGON exit routine can be used to determine the identity of the terminal for which the logon was generated. (See the EXLST macro instruction description.) If Q is set, a logon is generated when the first terminal becomes available. If NQ is set, and a terminal is available, the logon is generated immediately. If no terminals are available, no logon is generated.

ACCEPT | ACQUIRE

Indicates whether OPNDST is being issued to accept a terminal's logon or whether it is being issued to acquire that terminal.

ACCEPT

VTAM connects the application program to a terminal that has issued a logon. If the ANY option code is set and more than one terminal has issued a logon and is waiting to be accepted, the first terminal that issued a logon is connected. The symbolic name of that terminal is placed in the NIB pointed to by OPNDST's RPL. If the SPEC option code is in effect, the NIB must *already* contain the symbolic name of a terminal; connection is established only if that particular terminal issues a logon.

ACQUIRE

VTAM connects the application program to the terminal represented by this NIB if the terminal is available (that is, not connected to an application program and does not have a pending logon). The CONALL-CONANY option code determines which of the terminals represented in the list (that have not issued logons) are connected. If CONALL is in effect, all of the available terminals represented in the list are connected. If CONANY is in effect instead, only the first available terminal represented in that list is connected.

The use of ACQUIRE must be authorized for the application program by the user.

SPEC | ANY

When the RPL is used by an OPNDST macro with an ACCEPT option code, these option codes indicate the following:

SPEC

Connection is to be made to a specific terminal if that terminal issues (or has issued) a logon to the application program. The terminal is identified by placing its symbolic name in a NIB and placing the address of that NIB in the RPL's NIB field.

ANY

Connection is to be made to any terminal that has issued a logon for the application program.

When the RPL is used by a READ, SOLICIT, or RECEIVE macro instruction, these option codes indicate the following:

SPEC

Data is to be obtained from the specific terminal whose CID is in the RPL's ARG field.

ANY

For READ, data already obtained from any one terminal is to be moved to the application program's input area, subject to the setting of the terminal's CS-CA option code. For SOLICIT, data is to be obtained from all of the terminals connected to the application program, subject to the setting of the CS-CA option code. For RECEIVE, data arriving from any one logical unit is to be moved to the application program's input area, subject to the setting of the logical unit's CS-CA option code and the RTYPE field of the RECEIVE macro instruction.

QUIESCE | STOP | START
Indicates how a SETLOGON request is to affect (1) the queuing of logons for a given ACB and (2) the codes returned by INQUIRE (OPTCD=APPSTAT) issued by other application programs. This option code applies only if the ACB has been opened with MACRF=LOGON specified.

QUIESCE
No more logons can be directed at the ACB whose address is in the RPL's ACB field. Application programs issuing INQUIRE (OPTCD=APPSTAT) for the application program will receive a return code indicating that the application program cannot accept logon requests, presumably because it is about to close the ACB.

STOP
Application programs issuing INQUIRE (OPTCD=APPSTAT) for the ACB receive a return code indicating that no logons should be directed at the ACB (but implying that logons will be accepted later). The use of this option, however, does not prevent logons from being queued if the other application programs ignore this indicator and issue CLSDST (OPTCD=PASS) anyway. SETLOGON (OPTCD=STOP) should be used to temporarily halt logons; use SETLOGON (OPTCD=QUIESCE) to permanently bar logons to the ACB.

START
Application programs issuing INQUIRE (OPTCD=APPSTAT) receive a return code indicating that the application program represented by the ACB is accepting logons. This version of SETLOGON also causes VTAM to commence queuing automatic logon requests if this is the first such SETLOGON request since the ACB was opened. SETLOGON (OPTCD=START) reverses the effect of a previous SETLOGON (OPTCD=STOP).

RELEASE | PASS
Indicates whether or not a logon is to be generated when a CLSDST macro instruction is issued.

RELEASE
No logon is generated; the terminal is simply disconnected from the application program.

PASS
VTAM generates a simulated logon on behalf of the terminal being disconnected and directs this logon to the application program whose symbolic name is pointed to by the RPL's AAREA field. If the RPL's AREALEN field contains a value other than 0, VTAM also sends a logon message with the logon. VTAM obtains the message from the storage area identified in the AREA field, and sends the number of bytes indicated in the AREALEN field. The use of CLSDST with PASS must be authorized by the user.

LOGONMSG | DEVCHAR | COUNTS | TERMS | APPSTAT | CIDXLATE TOPLOGON | BSCID | SESSPARM
Indicates the action VTAM is to take when an INQUIRE macro instruction is issued.

LOGONMSG
INQUIRE retrieves the logon message of a terminal that has issued a logon for the application program.

The RPL's NIB field must point to a NIB whose NAME field contains the symbolic name of the terminal. The RPL's ACB field must indicate the ACB to which the logon was directed. This information is provided in the parameter list passed to the LOGON exit routine.

The AREA and AREALEN fields must indicate the location and length of the storage area where the logon message is to be placed.

DEVCHAR
INQUIRE obtains the device characteristics of a terminal, as they are defined by the user in the resource definition table at the time INQUIRE is executed. These device characteristics can be used by the application program to determine which processing options the program wants to set in the NIB used to connect the terminal.

The RPL's NIB field must point to a NIB containing the symbolic name of the terminal, or the RPL's ARG field must contain the CID of the terminal (see the description of the NIB field for more information on the NIB-ARG field). The device characteristics are placed in the program storage area whose location and length are indicated by the AREA and AREALEN fields of the RPL. See the INQUIRE macro instruction for details.

COUNTS
INQUIRE provides the number of terminals that are connected via a given ACB and the number of terminals that have logged on via that ACB but have not yet been connected. These two numbers are placed in a 4-byte area in program storage whose location and length are indicated by the AREA and AREALEN fields of the RPL. VTAM places the number of connected terminals in the first 2 bytes and the number of terminals requesting logon in the second 2 bytes.

The connections and logons counted by INQUIRE are those directed to the ACB indicated by the ACB field.

TERMS
When this operand is specified, node initialization blocks (NIBs) are built by INQUIRE.

The RPL's NIB field must point to a NIB whose NAME field contains the name of an entry that exists in the resource definition table at the time INQUIRE is issued. This entry must be a PU, LU, LINE, CLUSTER, or TERMINAL entry that represents several terminals. A NIB is built for each terminal represented in the entry.

Each generated NIB contains the symbolic name of the terminal. The flags for the LISTEND field are set to group the NIBs together into a NIB list. In addition, device characteristics are supplied in the DEVCHAR field of each NIB. These characteristics can be used to reset the processing options of the NIB to values that are appropriate for the terminal.

The user must set each NIB's MODE field to BASIC or RECORD before the NIBs are ready to be used for connection.

APPSTAT
This type of INQUIRE determines whether a given application program is available or unavailable. An available application program is one whose ACB is

active (open) and indicates that logons are to be accepted (OPEN with MACRF=LOGON and SETLOGON with OPTCD=START or STOP have been issued).

The RPL's NIB field must point to a NIB whose NAME field contains the symbolic name of the *application program* whose status is being checked. A value returned in the RPL's FDBK field indicates whether the application program is available or not. See the INQUIRE macro instruction description for the codes that can be returned.

CIDXLATE
INQUIRE provides the symbolic name of the terminal whose CID you provide, or provides the CID of the terminal whose symbolic name you provide.

If the RPL's ARG field contains the CID of the terminal, the 8-byte symbolic name of that terminal is returned in the data area indicated in the AREA field. If the RPL's NIB field contains the address of a NIB, the CID for the terminal whose symbolic name is in that NIB is placed in the RPL's ARG field, which replaces the NIB field.

TOPLOGON
For a given ACB, INQUIRE provides the symbolic name of the terminal that is currently at the top of the logon queue for that ACB (that is, the terminal that has spent the greatest amount of time waiting for its logon to be satisfied).

The RPL's ACB field must indicate the ACB whose logon queue is to be used. The 8-byte symbolic name of the terminal is returned in the data area indicated in the RPL's AREA field.

BSCID
INQUIRE returns the ID verification sequence of the terminal logging on. This form of INQUIRE is appropriate if the terminal's name (as provided in the LOGON exit routine's parameter list) is one of the names established during VTAM definition as an unidentified terminal with an ID verification feature.

The RPL's NIB field must point to a NIB whose NAME field contains the symbolic name provided in the LOGON parameter list. The sequence is placed in the work area defined by the AREA and AREALEN fields (set AREALEN to 20).

SESSPARM
INQUIRE returns the session parameters associated with a specified logon mode. The NIB field of the RPL must point to a NIB whose LOGMODE field identifies the logon mode to be used. The logon mode name that is specified in the NIB is used to search the logon mode table defined for the logical unit named in the NIB. If a match is found, the session parameters associated with the logon mode name are returned in the field pointed to by the AREA field of the RPL. INQUIRE may also be used to obtain the session parameters and logon data associated with a pending logon (character-coded logon, Initiate Self command, SIMLOGON macro, VARY LOGON operator command, or CLSDST (OPTCD= PASS) macro). The default session parameters may also be returned (that is, the first entry of the logon mode table defined for the logical unit). See the description of the LOGMODE operand of the NIB macro for more information.

SYN | ASY

Indicates whether VTAM should synchronously or asychronously handle any request made using this RPL.

SYN

Synchronous handling means that when a request is made, control is not returned to the application program until the requested operation has been completed (successfully or otherwise). The application program should not use the CHECK macro instruction for synchronous requests; VTAM automatically performs this checking (which includes clearing the internal ECB; an ECB address or an exit routine address should not be specified in the RPL or an RPL-based macro instruction should specify ECB=INTERNAL). When control is returned to the application program, registers 0 and 15 will contain completion codes.

ASY

Asynchronous handling means that after VTAM schedules the requested operation, control is immediately passed back to the application program. When the event has been completed, VTAM does one of the following:

- If the ECB address is specified for the RPL, VTAM posts a completion indicator in the event control block indicated by this operand. The application program must issue a CHECK (or a system WAIT) macro to determine whether the ECB has been posted. If an ECB or an EXIT address is specified in the RPL or an RPL-based macro instruction specifies ECB field itself as an internal event control block; the application program must issue a CHECK for the RPL to check this ECB.
- If the EXIT operand is in effect for the RPL, VTAM schedules the exit routine indicated by this operand. This exit routine should issue the CHECK macro so that the RPL can be reused, and also to cause automatic entry into a LERAD or SYNAD exit routine if the requested operation ends with a logical or other error. CHECK should be issued in the exit routine if the application program has no LERAD or SYNAD routine, since CHECK will return a code indicating whether or not a logical or other error occurred.

Note: *After an asynchronous request has been accepted and before that request has been completed, do not modify the RPL being used by the request. This restriction also applies to a NIB during OPNDST processing. A modification during this interval could cause VTAM to be unable to complete the request in a normal manner, which in turn would cause VTAM to terminate the application program.*

CA | CS

The CS (continue specific) and CA (continue any) option codes determine which type of input request is required to obtain data from the terminal.

CA

Places the terminal in a status wherein it's data is subject to RECEIVE or READ and SOLICIT macro instructions. This status is termed *continue-any mode.*

Although the CS-CA option code affects only RECEIVE, SOLICIT, or READ operations, you can switch a terminal from one status to the other by specifying the CS or CA option code in any OPNDST, SEND, RECEIVE, RESETSR, SOLICIT, READ, WRITE, DO, or RESET macro instruction. The change from one status to another is effective for the *next* I/O operation directed at the terminal, not when this macro instruction is executed. The terminal that is the object of the macro instruction is the one whose CS-CA status is changed. (For RECEIVE or READ with OPTCD=ANY, the terminal whose status will be changed is the one whose data is moved by the READ or RECEIVE operation.) If an error occurs and a macro instruction that specifies a change in a terminal's CA-CS mode is not completed successfully, the CS-CS mode of the terminal is not changed.

CS
Places the terminal into a status wherein only input requests that are directed *specifically* at the terminal can be used to obtain data from the terminal. These are the RECEIVE, READ, and SOLICIT macro instructions with OPTCD=SPEC specified. Looking at CS another way, it "immunizes" a terminal from input requests that are *not* specifically directed at the terminal–namely, RECEIVE, READ, or SOLICIT macro instructions issued with OPTCD=ANY. The status into which the terminal is placed is termed *continue-specific mode.*

For example, while CS is in effect, the arrival of data from a logical unit that is in continue-specific mode does not trigger the completion of a RECEIVE (OPTCD=ANY) macro instruction that may have already been issued.

Continue-any and continue-specific modes can be set individually for a particular type of terminal input. For example, a terminal can be placed in continue-specific mode for normal-flow messages while it is in continue-any mode for expedited-flow messages and for responses.

BLK | LBM | LBT (Basic mode only)
Indicates that the block of data to be transferred on a WRITE operation represents a block (BLK), the last block of a message (LBM), or the last block of a transmission (LBT). Appendix B shows the line-control characters sent when each of these three option codes are in effect. BLK is invalid for a 3270 Information Display System.

NCONV| CONV (Basic mode only)
Indicates whether or not a WRITE macro instruction is to be handled as a conversational write request.

NCONV
Only the output operation is performed.

CONV
Following the output operation, data is obtained from the terminal and placed in the area in program storage indicated by the RPL's AAREA field.

COND | UNCOND | LOCK (Basic mode only)
Indicates the action to be taken when a RESET macro instruction is issued.

COND
RESET cancels any I/O operation that is pending for a specific terminal, but does not affect an I/O operation if data transfer has begun. This form of RESET cannot be used if an error lock is set.

UNCOND
RESET cancels any I/O operation with a specific terminal, whether or not data transfer has begun. Any data that has already been brought into VTAM buffers

is kept by VTAM for subsequent retrieval by the application program (with a READ macro). Any data being sent or about to be sent by the terminal may be lost. RESET also resets any error lock that has been set for the terminal.

LOCK
RESET resets any error lock that has been set for the terminal.

NERASE | ERASE | EAU (Basic mode only)
Indicates the action to be taken when a WRITE macro instruction is issued.

NERASE
WRITE performs an ordinary write operation with no display screen erasure. Use this option for all devices other than 3270 and 2770 devices.

ERASE
WRITE erases the screen of a display device attached to a 3270 Information Display System or a 2770 Data Communication System, and then sends a block of data to the device.

EAU
WRITE erases only the unprotected portion of the screen of a display device attached to a 3270 Information Display System. No data is written.

RELRQ | NRELRQ
Indicates the action to be taken when a SIMLOGON macro is issued, and the terminal that is the object of this simulated logon request is already connected to another application program–that is, already connected to an ACB other than the one being used for the SIMLOGON macro.

The effect of this option is to determine whether or not the application program that is connected to the terminal is to be notified of your request. The NRELRQ option, for example, allows you to release a terminal to another application program, and then immediately request reconnection to assure its eventual return to your program without notifying the receiving application program.

Note the difference in spelling between the RELRQ-NRELRQ RPL option, and the related exit routine. The latter is coded in the EXLST macro instruction as RELREQ.

RELRQ
If the application program to which the terminal is connected has a RELREQ exit routine, that routine is invoked.

NRELRQ
No RELREQ exit routine is invoked.

Q | NQ
Indicates the action VTAM is to take when the application program issues SIMLOGON or OPNDST (ACCEPT) and the terminal that is the object of this request is unavailable.

Q
For SIMLOGON, VTAM is to schedule the LOGON exit routine when the terminal is finally available and complete the SIMLOGON request when it has

done so. For OPNDST with ACCEPT, VTAM completes the OPNDST when the terminal logs onto this application program. If a logon is already queued, the OPNDST is completed immediately.

NQ
VTAM is to immediately return control to the application program. (Without the NQ option, a connection request or simulated logon might remain pending indefinitely, until another application program releases the terminal.)

The Q-NQ option also indicates the action VTAM is to take when the application program issues a RECEIVE or RCVCMD macro instruction and no input that is eligible to satisfy the request is at that moment in VTAM's buffers.

Q
VTAM is to satisfy the request when the input is finally available and complete the RECEIVE or RCVCMD when it has done so.

NQ
VTAM is to terminate the request and return control to the application program immediately without performing any CA-CS mode switching.

RPL Fields Set by VTAM

All of the RPL fields described above are fields that are set by the application program. The following fields are set by VTAM. Some of the fields described below are initially set by the application program and are then (for certain macro instructions) reset by VTAM before the macro instruction is completed. Figure 10 identifies the RPL fields that are so used.

In some cases, fields set by VTAM prior to the completion of one macro instruction will cause erroneous results if the application reuses the same RPL for another macro without again initializing the field. (Only the SSENSEI, SSENSMI, USENSEI, SSENSEO, SSENSMO, USENSEO, FDBK, FDBK2, RTNCD, and SENSE fields are cleared by VTAM, and no fields are reset to their original values by VTAM.)

For example: Before a RECEIVE is issued, the RTYPE field is set by the application program to indicate the types of input (DFSYN, DFASY, RESP) that are eligible to satisfy the RECEIVE. The application program might indicate all 3. When the RECEIVE is completed, VTAM uses the same field to indicate the type of input that actually satisfied the RECEIVE; if a response was received, for instance, VTAM would reset the RTYPE field to RTYPE=(NDFSYN, NDFASY, RESP). Should the application program issue another RECEIVE with the same RPL and fail to reset the RTYPE field to its intended setting, the second RECEIVE could only receive responses.

Field Name	Content
ARECLEN	The number of bytes of data returned by the WRITE (OPTCD=CONV) and INTRPRET macro instructions. See WRITE and INTRPRET for details.
RTNCD	A general return code returned by all of the RPL-based macro instructions. This field is cleared by VTAM when the processing of the macro instruction begins. This is one of the feedback fields described in Appendix C.

FDBK2 — A specific error return code returned by all RPL-based macro instructions that are accepted by VTAM but are not completed successfully. This field is cleared by VTAM when the processing of the macro instruction begins. This is one of the feedback fields described in Appendix C. A DSECT containing labeled EQU instructions for each FDBK2 return code is described in Appendix H (ISTUSFBC). These DSECT labels can be used instead of the numerical values that are cited for FDBK2 throughout this manual.

FDBK — Status information for INQUIRE, READ, conversational WRITE, or DO macro instructions. For example, if the data ended with an EOM line-control character, this field is set to indicate this. This field is cleared by VTAM when the processing of the macro instruction begins. This is one of the feedback fields described in Appendix C.

SENSE — The SENSE field contains status or sense bytes obtained from certain devices. The SENSE field applies only to DO, READ, and WRITE macro instructions, and is set following these macro instructions only if the RPL's FDBK2 field so indicates. This field is cleared by VTAM when the processing of the macro instruction begins. There is more information about the SENSE field in Appendix C.

REQ — A value returned by all RPL-based macro instructions except EXECRPL (and CHECK) that identifies the type of macro instruction. This field shows which type of macro instruction last used the RPL. These are the values that are set:

Value Hex (Dec)	**Macro Instruction**
11 (17)	WRITE
12 (18)	RESET
13 (19)	DO
15 (21)	SETLOGON
16 (22)	SIMLOGON
17 (23)	OPNDST
19 (25)	CHANGE
1A (26)	INQUIRE
1B (27)	INTRPRET
1D (29)	READ
1E (30)	SOLICIT
1F (31)	CLSDST
22 (34)	SEND
23 (35)	RECEIVE
24 (36)	RESETSR
25 (37)	SESSIONC
27 (39)	SENDCMD
28 (40)	RCVCMD

USER — Upon the completion of a SEND, RECEIVE, RESETSR, SESSIONC, READ, WRITE, SOLICIT, RESET, or DO macro instruction, this field contains whatever value you previously placed in the USERFLD field of the NIB used to connect the terminal. See the description of the USERFLD operand of the NIB macro instruction for more information.

ARG	A terminal's communication identifier (CID) is provided by OPNDST. The CID is a 4-byte network-oriented version of the terminal's symbolic name. It is generated by VTAM when the terminal is connected to the application program, and is used by the application program to indicate identity of the terminals for subsequent I/O requests. More information about the CID and its use appears above in the description of the NIB operand of this macro instruction.
ECB/EXIT	If this field is used as an internal ECB (that is, if neither ECB or EXIT is specified or if an RPL-based macro specifies ECB=INTERNAL), VTAM sets and clears it. If synchronous requests are issued (OPTCD=SYN) and EXIT is specified, VTAM ignores the address specified and uses this field as an internal ECB.
AREA	After an OPNDST macro instruction is issued, AREA is the address of a NIB or a list of NIBs. (The NIB field is replaced by the CID.) If the RPL for the OPNDST is to be reused for subsequent I/O operations, the AREA field must be reset to indicate the data area to be used for the I/O operation.
AAREA	After a DO macro instruction is completed, the AAREA field is set to the address of the last LDO.
RECLEN	After an INQUIRE macro instruction is completed RECLEN contains the length of the requested information (such as the logon message). After a SETLOGON macro instruction is completed, RECLEN contains the number of logon requests already queued for the application program. After a READ, DO, or RCVCMD macro instruction is completed, RECLEN contains the amount of input data. After a RECEIVE, RECLEN contains the total length of data available in VTAM's buffer. This value may be greater than AREALEN, indicating the presence of excess data (the value in RECLEN represents the total length of excess data plus the data in AREA).
CONTROL	After a RECEIVE macro instruction (RTYPE=DFSYN) is completed, CONTROL is set to one of the following values:

CONTROL=DATA	(data message received)
CONTROL=QC	(Quiesce Complete command received)
CONTROL=CANCEL	(Cancel command received)
CONTROL=CHASE	(Chase command received)
CONTROL=LUS	(Logical Unit Status command received; check SSENSEI, SSENSMI, and USENSEI)
CONTROL=RTR	(Ready to Receive command received)

After a RECEIVE macro instruction (RTYPE=DFASY) is completed, CONTROL is set to one of the following values:

CONTROL=QEC	(Quiesce at End of Chain command received)
CONTROL=RELQ	(Release Quiesce command received)
CONTROL=SHUTC	(Shutdown Complete command received)
CONTROL=RSHUTD	(Request Shutdown command received)
CONTROL=SIGNAL	(Signal command received; check SIGDATA)

After a SCIP exit routine is entered, the CONTROL field of the read-only RPL is set to the following value:

CONTROL=RQR	(Request Recovery command received)

SEQNO When a SEND (POST=RESP) or a RECEIVE macro instruction has received a response, the SEQNO field contains the sequence number of the message being responded to. When a SEND (POST=SCHED) is used to send a message (STYPE=REQ), the SEQNO field contains the VTAM-supplied sequence number of the message.

RESPOND When a SEND (POST=RESP) or a RECEIVE macro instruction has received a response, the RESPOND field indicates whether the response is a positive response (NEX) or a negative response (EX)—and the type of the response—response type 1 (FME), response type 2 (RRN), or both (FME, RRN). When a RECEIVE macro instruction has received a message, the RESPOND field indicates the expected response—definite (NEX) or exception (EX)—and the type of the expected response—response type 1 (FME), response type 2 (RRN), or both (FME, RRN).

CHAIN When a RECEIVE (RTYPE-DFSYN) macro instruction has received a message, the CHAIN field indicates the message's relative position within the chain. The possible settings are CHAIN=FIRST, CHAIN=MIDDLE, CHAIN=LAST, and CHAIN=ONLY.

CHNGDIR When a message or response is received, the CHNGDIR field indicates the presence of change-direction indicators. The possible CHNGDIR settings are:

CHNGDIR=(NCMD,REQ)	(Change Direction Request indicator present—DFASY or RESP only)
CHNGDIR=(CMD,NREQ)	(Change Direction Command indicator present—DFSYN only)
CHNGDIR=(NCMD,NREQ)	(neither indicator present)

BRACKET When a message is received, the BRACKET field indicates the presense of bracket indicators. The possible BRACKET settings are:

BRACKET=(BB,EB)	(Begin Bracket and End Bracket indicators both present)
BRACKET=(NBB,EB)	(End Bracket indicator present)
BRACKET=(BB,NBB)	(Begin Bracket indicator present)
BRACKET=(NBB,NEB)	(neither indicator present)

RTYPE When a RECEIVE macro instruction is completed, the RTYPE field indicates the type of input that caused the completion. The possible RTYPE field settings are:

RTYPE=DFSYN	(data or other normal-flow message received)
RTYPE=DFASY	(expedited-flow message received)
RTYPE=RESP	(response received)

SSENSEI When a SEND (POST=RESP), RECEIVE, or SESSIONC macro instruction receives an exception message or response, the SSENSEI field indicates the presence of a system sense error code. The SSENSEI field may also contain a system sense error code upon completion of an OPNDST for a logical unit. This field is cleared by VTAM when the processing of the macro instruction begins. These codes are described in Appendix C. Possible SSENSEI settings are:

SSENSEI=PATH	(unrecoverable path error condition)
SSENSEI=CPM	(request header error condition)
SSENSEI=STATE	(state error condition)
SSENSEI=FI	(request error condition)
SSENSEI=RR	(request reject condition)
SSENSEI=0	(no system sense error code)

SSENSMI — When a SEND (POST=RESP), RECEIVE, or SESSIONC macro instruction receives an exception message or response, the SSENSMI field indicates the presence of a system sense modifier value. The SSENSMI field may also contain a system sense modifier value upon completion of an OPNDST for a logical unit. The modifier values are described in Appendix C. The value is tested as a 1-byte quantity. If SHOWCB is used, the value is right-adjusted in the 4-byte work area; the other 3 bytes are set to 0. This field is cleared by VTAM when the processing of the macro instruction begins.

USENSEI — When a SEND (POST=RESP), RECEIVE, or SESSIONC macro instruction receives an exception message or response, the USENSEI field may contain a user sense value or status/sense information returned for non-SNA 3270 devices (see Appendix I). The USENSEI field may also contain a user sense value upon completion of an OPNDST for a logical unit. This value is tested as a 2-byte quantity. If SHOWCB is used, the value is right-adjusted in the 4-byte work area; the other 2 bytes are set to 0. This field is cleared by VTAM when the processing of the macro instruction begins.

SSENSEO — This field is always set to 0 when an RPL-based macro is completed.

SSENSMO — This field is always set to 0 when an RPL-based macro is completed.

USENSEO — This field is always set to 0 when an RPL-based macro is completed.

IBSQAC — When a SESSIONC macro instruction (CONTROL=STSN) is completed, the IBSQAC field contains the logical unit's response regarding the inbound sequence number. Possible settings are TESTPOS, TESTNEG, INVALID, and RESET. See the SESSIONC macro instruction for more information.

OBSQAC — When a SESSION macro instruction (CONTROL=STSN) is completed, the OBSQAC field contains the logical unit's response regarding the outbound sequence number. The possible settings for OBSQAC are the same as those for IBSQAC.

IBSQVAL — When a SESSIONC macro instruction (CONTROL=STSN) is completed and IBSQAC is set to TESTPOS or TESTNEG, the IBSQVAL field contains the logical unit's version of the inbound sequence number.

OBSQVAL	When a SESSIONC macro instruction (CONTROL=STSN) is completed and OBSQAC is set TESTPOS or TESTNEG, the OBSQVAL field contains the logical unit's version of the outbound sequence number.
SIGDATA	When a SIGNAL command is received, the SIGDATA field contains the signal information sent by the terminal. The value in this field is tested as a 4-byte quantity.
OPTCD	When a RECEIVE macro instruction is completed and a function management header is present, OPTCD field is set to indicate FMHDR. The OPTCD field is also set when SNA data flow control or function management data requests or responses are received. The field thus corresponds to the presence of the format indicator in the SNA request header of the received request or response.
CODESEL	When a RECEIVE macro instruction is completed, the CODESEL field indicates whether the data received is encoded in the previously agreed upon standard code (STANDARD) or some other code (ALT).

Examples

```
RPL1        RPL         ACB=ACB1,NIB=NIB1,AM=VTAM,
                        OPTCD=(SPEC,ASY),
                        EXIT=EXITPGM
```

RPL1 can be used by an OPNDST macro instruction to connect the terminal represented in NIB1 to the application program, that is, to ACB1. When the operation is completed, the application program will be interrupted, and the routine at EXITPGM is invoked.

```
RPL2        RPL         ACB=ACB1,AM=VTAM,AREA=SOURCE,
                        POST=RESP,RECLEN=132,ECB=ECBWORD,
                        OPTCD=ASY
```

RPL2 can be used by a SEND macro instruction to write a data message (132 bytes from SOURCE) to a terminal. When the request has been accepted, control is returned. When the request has been completed, ECBWORD is posted. (The CHECK macro instruction used to check the send operation would point to RPL2.)

RPL Fields and RPL-Based Macro Instructions

Figure 11 shows all of the VTAM macro instructions that allow RPL modifications to be indicated when the macro instruction is coded. It also shows all of the RPL fields, including the option codes of the OPTCD field, that might be modified by the application program or by VTAM. The symbols in the table indicate, for a given macro instruction, the RPL fields that are set by VTAM or by the application program.

The programmer coding the macro should be aware of that field's effect either by checking the the description of that macro instruction or by checking the field's description in the RPL macro instruction. The absence of an A or V means that the contents of that field can be safely ignored when that macro instruction is issued.

Note: *When issuing an RPL-based macro instruction, the value remaining in an RPL field after its last use is reused unless the application program explicitly alters the field. There are, therefore, no default values for the operands of RPL-based macro instructions.*

RPL—modifying macro instructions →		CLSDST						OPNDST													
Applicable RPL fields:	CHANGE	(PASS)	(RELEASE)	DO	EXECRPL	INQUIRE	INTRPRET	(ACQUIRE)	(ACCEPT)	RCVCMD	READ	RECEIVE	RESET	RESETSR	SEND	SENDCMD	SESSIONC	SETLOGON	SIMLOGON	SOLICIT	WRITE
ACB	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
ARG/NIB (When ARG specified)		A	A	A	AV	A	A	V	V		AV	AV	A	A	A		A			A	A
ARG/NIB (When NIB specified)	A	A	A		A	A	A	A	A										A		
AREA		A		A	AV	A	A	V		A	A	A			A	A			A		A
AREALEN					A	A				A	A	A									
RECLEN		A		V	AV	V	A			V	V	V			A	A		V	A		A
AAREA		A		V	AV		A														A
AAREALN					A		A														A
ARECLEN					V		V														V
BRANCH					A							A		A	A						
EXIT/ECB (When ECB specified)	For all macros: A																				
EXIT/ECB (When EXIT specified)	For all macros: If OPTCD=ASY, A; if OPTCD=SYN, AV																				
EXIT/ECB (When neither is specified)	For all macros: V																				
REQ	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
RTNCD[1]	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
FDBK2[1]	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
FDBK[1]				V	V	V					V										V
SENSE[1]				V	V						V									V	
USER				V	V						V	V	V	V	V		V			V	V
SEQNO					AV							V			AV						
POST					A										A						
RESPOND					AV							V			AV						
CONTROL					AV							V			A		A				
CHAIN					AV							V			AV[1]						
CHNGDIR					AV							V			AV[1]						
BRACKET					AV							V			A						
RTYPE					AV							AV		A	AV[1]						
STYPE					A										A		A				
SSENSEO[2]					AV										AV						
SSENSMO[2]					AV										AV						
USENSEO[2]					AV										AV						
SSENSEI[1]					V			V	V			V			V		V				
SSENSMI[1]					V			V	V			V			V		V				
USENSEI[1]					V			V	V			V			V		V				
IBSQAC					AV												AV				

Figure 11 (Part 1 of 2). The RPL Fields Applicable to the Macro Instructions that Can Modify RPLs

RPL–modifying macro instructions → Applicable RPL fields:	CHANGE	CLSDST (PASS)	CLSDST (RELEASE)	DO	EXECRPL	INQUIRE	INTRPRET	OPNDST (ACQUIRE)	OPNDST (ACCEPT)	RCVCMD	READ	RECEIVE	RESET	RESETSR	SEND	SENDCMD	SESSIONC	SETLOGON	SIMLOGON	SOLICIT	WRITE
OBSQAC					AV												AV				
IBSQVAL					AV												AV				
OBSQVAL					AV												AV				
SIGDATA					AV							V			A						
CODESEL					AV							V			A						
OPTCD:																					
TRUNC–KEEP–NIBTK					A							A									
FMHDR–NFMHDR					AV							V			A						
CONANY–CONALL					A			A											A		
ACQUIRE–ACCEPT					A			A	A												
SPEC–ANY					A				A		A	A								A	
QUIESCE–START–STOP					A													A			
PASS–RELEASE		A	A		A																
LOGONMSG–DEVCHAR–COUNTS–BSCID–TERMS–APPSTAT–CIDXLATE–TOPLOGON–SESSPARM					A	A															
SYN–ASY	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
CS–CA	A			A	A			A	A		A	A	A	A	A					A	A
BLK–LBM–LBT					A																A
CONV–NCONV					A																A
COND–UNCOND–LOCK					A								A								
ERASE–EAU–NERASE					A																A
RELRQ–NRELRQ					A														A		
Q–NQ					A				A	A		A							A		

1 When set by VTAM, these fields are cleared (set to 0) when the processing of the macro instruction begins.

2 These fields are cleared by VTAM on completion of all RPL-based macros that have been accepted.

The presence of a symbol means that the RPL field or option code is applicable for the macro instruction in one of three ways:

$\boxed{A}$ The field or option code is set by the *application program* to supply VTAM information about the request.

$\boxed{V}$ The field is set by *VTAM* when the request has been completely processed.

$\boxed{AV}$ The field is set by the *application program* and, then reset by *VTAM*. Users intending to reissue requests that use these fields should reinitialize them first.
See the restriction that appears in the EXECRPL macro instruction description and in the RPL macro instruction description under "RPL Fields Set by VTAM".

Figure 11 (Part 2 of 2). The RPL Fields Applicable to the Macro Instructions that Can Modify RPLs

SEND–Send Output to a Logical Unit (Record Mode Only)

The SEND macro instruction transmits a message or a response to a logical unit or to a BSC or local non-SNA 3270 terminal used in record-mode. The major options for a SEND macro instruction are illustrated in Figure 12.

Two options are available when data or other normal-flow messages are sent. The first, scheduled output, is completed as soon as the output data area containing the message can be reused. This occurs prior to the actual transmission of the message. The resulting response (if any) is received using a separate RECEIVE macro instruction or in a RESP exit routine. The second option, responded output, is not completed until the message has been transmitted and a response is returned. The RPL used for the SEND macro instruction is used to receive the response; no separate RECEIVE macro instruction or RESP exit routine is used. Responded output can be used only when a response is assured.

The RESPLIM field of a logical unit's NIB limits the number of concurrent responded output requests. Scheduled output requests cannot be stacked in this manner, however. Only one outstanding (uncompleted) scheduled output request is permitted at a time.

Name	*Operation*	*Operands*
[symbol]	**SEND**	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address
Indicates the location of the RPL that describes the SEND operation.

rpl field name=new value
Indicates an RPL field to be modified, and the new value that is to be contained or represented within it. To avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the SEND macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register. ARG=(register) can also be coded.

The following RPL operands apply to the SEND macro instruction:

ACB=acb address
Indicates the ACB that identifies the application program and was used when the receiving logical unit was connected.

ARG=(register)
The SEND macro instruction is always directed at one specific logical unit. The ARG operand specifies the register containing the CID of the logical unit being transmitted to. If the ARG field is not modified, the CID that is already in the RPL's ARG field is used.

The SEND macro instruction is used in three major ways:

Send a Data Message
STYPE = REQ
CONTROL = DATA

Responded output:	The send operation will be completed when a response to this data is received from the logical unit POST = RESP
Scheduled output:	The send operation will be completed as soon as the output data area is free; a response (if any) will be obtained with a RECEIVE macro instruction or a RESP exit routine POST = SCHED

Send a Nondata Message.
STYPE=REQ,
CONTROL=

QEC	Send a Quiesce at End of Chain Command
RELQ	Send a Release Quiesce Command
QC	Send a Quiesce Complete Command
CANCEL	Send a Cancel Command
CHASE	Send a Chase Command
BID	Send a Bid Command
SHUTD	Send a Shutdown Command
LUS	Send a Logical Unit Status Command
SIGNAL	Send a Signal Command

Send a Response to a data message or to a Cancel, Chase, Logical Unit Status or Quiesce Complete command.
STYPE=RESP,CONTROL=DATA|CANCEL|CHASE|LUS|QC*

Positive Response	
Definite response 1	Definite response 2
Definite responses 1 and 2	

Negative Response	
Definite response 1	Definite response 2
Definite responses 1 and 2	

*Note: RTR cannot be set by the application program; use the RPL that was used to receive the RTR command.

Figure 12. The Major SEND Options

AREA=output data address

The data contained at the location designated by AREA is sent to the logical unit. This storage can be reused as soon as VTAM has transferred the data to its own buffers (see POST=SCHED below). This operand is meaningful only if data is being sent (CONTROL=DATA).

RECLEN=output data length

The number of bytes of data indicated in this field is sent to the logical unit. The maximum value that can be set is 32767. If the RECLEN field is set to 0, the AREA field is not examined.

If the length specified in the RECLEN field exceeds the length of the data to be sent by the application program, the content of the data received at the logical unit is unpredictable.

STYPE=REQ | RESP

Designates whether a message (STYPE=REQ) or a response (STYPE=RESP) is to be sent. The CONTROL field governs the type of message sent, while the RESPOND field governs the type of response sent.

CONTROL=DATA | QEC | RELQ | QC | CANCEL | CHASE | SHUTD | BID LUS | SIGNAL

CONTROL=DATA

Sends a data message.

CONTROL=QEC

Sends a Quiesce at End of Chain command. This informs the logical unit that when it is through transmitting the current chain it is to stop transmitting further chains and return a Quiesce Completed (QC) command.

CONTROL=RELQ

Sends a Release Quiesce command. This informs the logical unit that it can begin transmitting messages.

CONTROL=QC

Sends a Quiesce Complete command. This informs the logical unit that the application program will no longer transmit normal-flow requests. Once this command is sent, no normal-flow requests can be transmitted to the logical unit until the logical unit returns a Release Quiesce (RELQ) command. The only request that can be sent in the quiesce state is the Quiesce at End of Chain (QEC) command.

CONTROL=CANCEL

Sends a Cancel command. The logical unit may interpret this signal as an indication that the logical unit should discard the chain that it is receiving. A Cancel command is sent instead of indicating that a message is the last message (CHAIN=LAST) when a message chain is canceled.

CONTROL=CHASE

Sends a Chase command. When the application program receives a response to this command, it can be certain that no messages are in the network for which the logical unit has failed to return a response.

CONTROL=SHUTD
Sends a Shutdown command. The logical unit interprets this as an indication that the application program is about to disconnect the logical unit. When the logical unit is ready to accept disconnection, it returns a Shutdown Complete (SHUTC) command.

CONTROL=BID
Sends a Bid command. The logical unit interprets this as a request on the part of the application program for permission to begin a new bracket.

CONTROL=LUS
Sends a Logical Unit Status (LUS) command. An LUS command conveys exactly the same type of information as does a negative response. An LUS is sent when the application program wishes to raise an exception condition, but cannot do so with an exception response (for example, the logical unit is sending messages and requesting no responses whatever). The SSENSEO, SSENSMO, and USENSEO fields are used for LUS indicators.

CONTROL=SIGNAL
Sends the 4 bytes of signal data contained in the SIGDATA field of the RPL.

BRACKET=(BB|NBB,EB|NEB)
This field indicates whether the message forms the beginning, middle, end, or sole element of a bracket. This operand is meaningful only when bracket protocol is being used by the logical unit (see *VTAM Concepts and Planning* for an explanation of bracket protocol).

BRACKET=(BB,NEB)
This is the beginning of a bracket.

BRACKET=(NBB,NEB)
This is the middle of a bracket.

BRACKET=(NBB,EB)
This is the end of a bracket.

BRACKET=(BB,EB)
This message is a bracket.

CODESEL=STANDARD|ALT

This operand indicates the type of message code. A previously determined standard (STANDARD), (for example, EBCDIC), may be specified or an alternate code (ALT), (for example, ASCII) may be specified.

CHNGDIR=(CMD|NCMD,REQ|NREQ)
This operand indicates the type of change-direction indicator to be sent (see *VTAM Concepts and Planning* for an explanation of change-direction indicators):

CHNGDIR=(CMD,NREQ)
Send a Change Direction Command indicator (valid only for DFSYN).

CHNGDIR=(NCMD,REQ)
Send a Change Direction Request indicator (valid only for RESP and DFASY).

CHNGDIR=(NCMD,NREQ)
Send no change-direction indicators.

CHAIN=FIRST|MIDDLE|LAST|ONLY
Indicates the position of the message within the chain currently being transmitted.

BRANCH=YES|NO
If SEND is being issued in an application program that is running in privileged state under a TCB (OS/VS2 MVS only), BRANCH can be set to YES. See the RPL macro instruction for more information.

POST=SCHED|RESP
This operand defines at what point the SEND operation is to be completed. (The OPTCD=SYN|ASY, ECB, and EXIT operands govern the action to be taken when the macro instruction is completed.) The POST operand applies only to the transmission of data messages.

POST=SCHED (scheduled output)
Indicates that the SEND operation is completed as soon as the output data area (pointed to by the AREA field) and the RPL are available for reuse. This occurs prior to the actual transmission of the data from the CPU. A RECEIVE macro instruction (or a RESP exit routine) is required to obtain the response. Only one SEND with POST=SCHED can be outstanding for a given logical unit at one time. A second SEND with POST=SCHED issued before the first has been completed is rejected by VTAM with a logical error return code. POST=SCHED is assumed if the RESPOND field indicates that no response (or only an exception response) is expected—that is, if a response is not assured.

POST=RESP (responded output)
Indicates that the SEND operation is completed when a response is returned from the logical unit. A RECEIVE is not used to obtain the message's response; the response information is posted in the SEND RPL. The RESPLIM field of the logical unit's NIB limits the number of SEND macro instructions with POST=RESP that can be outstanding at one time. POST=RESP can only be used when a response is assured—that is, when the RESPOND field of the SEND RPL is set to NEX. If EX is used for RESPOND, POST=SCHED is assumed by VTAM (the actual setting of the POST field is ignored).

When a response is being sent by the application program (STYPE=RESP) posting takes place as though POST=SCHED had been specified (the actual setting of the POST field is ignored). The limit of one SEND (POST=SCHED) outstanding for a logical unit at a time does not apply to the sending of responses.

When a nondata message is sent (STYPE=REQ, CONTROL set to other than DATA), posting takes place as though POST=RESP had been specified; the actual setting of the POST field is ignored.

The following table summarizes the use of the POST operand:

If ... STYPE=	CONTROL=	RESPOND=	Then ... POST=
REQ	DATA	NEX	SCHED or RESP is valid
REQ	DATA	EX	SCHED is assumed*
REQ	Anything except DATA		RESP is assumed*
RESP			SCHED is assumed*

*The actual value specified is ignored.

SIGDATA=signal data
Contains the signal data that is to be sent when CONTROL=SIGNAL is specified. The signal data can be a decimal, hexadecimal, or character constant of from 1 to 4 bytes or a register (the value in the register is used).

ECB=ecb exit routine address|INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) SEND operation is completed. The actual or implied setting of the POST field governs what constitutes the "completion" of the SEND operation. If EXIT is specified, the RPL exit routine is scheduled. Otherwise, the ECB is posted and CHECK or WAIT must be used to determine when posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN|ASY
When SYN is set, control is returned to the application program when the SEND operation is completed (see the POST operand above). When ASY is set, control is returned as soon as VTAM has accepted the SEND request. Once the operation has been completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB-EXIT field. See the RPL macro instruction for more information.

OPTCD=CA|CS
When the SEND operation is completed, the logical unit is placed in continue-any mode (OPTCD=CA) or continue-specific mode (OPTCD=CS).

The switch of continue-any and continue-specific modes applies to the type of input specified by the RTYPE field. More than one type of input can be specified. VTAM switches the modes for all specified types of input. No switching occurs if RTYPE=(NDFSYN,NDFASY,NRESP) is in effect for the SEND.

RTYPE=(DFSYN|NDFSYN,DFASY|NDFASY,RESP|NRESP)
When a SEND is issued, the RTYPE field indicates the type or types of input for which the logical unit's CA-CS mode is to be switched.

RTYPE=DFSYN|NDFSYN
When DFSYN is specified, the logical unit's CA-CS mode is switched for normal-flow messages. These include data and the Quiesce Complete, Cancel, Chase, Bid, Logical Unit Status, and Ready to Receive commands.

RTYPE=DFASY|NDFASY
When DFASY is specified, the logical unit's CA-CS mode is switched for expedited-flow messages. These include the Quiesce at End of Chain, Release Quiesce, Shutdown, Request Shutdown, Shutdown Complete, and Signal commands.

RTYPE=RESP|NRESP
When RESP is specified, the logical unit's CA-CS mode is switched for normal-flow responses.

OPTCD=FMHDR|NFMHDR
When OPTCD=FMHDR is used, the receiver is notified that the data contains a function management header.

SEQNO=sequence number
This field is set by the application program only when a response is being sent (STYPE=RESP). The sequence number is the same as the sequence number of a message to which a response is required but to which you have not yet responded. If the RPL used to receive a message is used to send the response, the SEQNO field will already be set to the correct value.

RESPOND=([EX|NEX] [,FME|NFME] [,RRN|NRRN])
When a message is being sent (STYPE=REQ), this field indicates the requested response:

RESPOND=(EX,FME,RRN)
Only negative responses type 1 or 2 are expected (see note below).

RESPOND=(EX,FME,NRRN)
Only a negative response type 1 is expected.

RESPOND=(EX,NFME,RRN)
Only a negative response type 2 is expected.

RESPOND=(EX,NFME,NRRN)
No response expected.

RESPOND=(NEX,FME,RRN)
Definite response type 1 or 2 is expected (see note below).

RESPOND=(NEX,FME,NRRN)
Definite response type 1 is expected.

RESPOND=(NEX,NFME,RRN)
Definite response type 2 is expected.

RESPOND=(NEX,NFME,NRRN)
No response expected.

Note: *When both definite responses 1 and 2 are returned and POST=RESP for the SEND RPL, the first response completes the SEND request. If the two responses are returned together, both are received as one response—that is, the second response is also reflected in the completed RPL. If the second response does not accompany the first, however, the second response must be received by a separate RECEIVE macro instruction or by a RESP exit routine.*

When a response is being sent (STYPE=RESP), this field indicates the response type:

RESPOND=(EX,FME,RRN)
This is a negative response type 1 or 2.

RESPOND=(EX,FME,NRRN)
This is a negative response type 1.

RESPOND=(EX,NFME,RRN)
This is a negative response type 2.

RESPOND=(EX,NFME,NRRN)
Invalid.

RESPOND=(NEX,FME,RRN)
This is a positive response type 1 or 2.

RESPOND=(NEX,FME,NRRN)
This is a positive response type 1.

RESPOND=(NEX,NFME,RRN)
This is a positive response type 2.

RESPOND=(NEX,NFME,NRRN)
Invalid.

Note: *Although VTAM and certain logical units permit response types 1 and 2 to be sent separately, this does not correspond to SNA requirements.*

SSENSEO=CPM | STATE | FI | RR | 0
This field contains the system sense code that is to be sent to the logical unit as part of a negative response or Logical Unit Status (LUS) command. The system sense code represents a major class of error and is used in conjunction with the SSENSMO (system sense modifier value) to describe a specific type of error. There is more information about the SSENSEO and SSENSMO fields near the end of Appendix C.

Note: *When an RPL is assembled or generated, and each time the RPL is reset to its inactive state (that is, after each synchronous request with OPTCD=SYN or CHECK macro instruction), the SSENSEO field is cleared.*

SSENSMO=system sense modifier value
This field, in conjunction with the code in the SSENSEO field, defines a particular type of SNA-defined error. The type of error represented by each system sense modifier value is described near the end of Appendix C. Like SSENSEO, SSENSMO is meaningful only when a negative response or a Logical Unit Status (LUS) command is sent to the logical unit. The system sense modifier value is coded as a decimal quantity or as any 1-byte framed hexadecimal (such as SSENSMO=X‘1B’ as defined in Appendix C). Register notation can also be used.

Note: *When an RPL is assembled or generated, and each time the RPL is reset to its inactive state (that is, after each synchronous request with OPTCD=SYN or CHECK macro instruction), the SSENSMO field is set to 0.*

USENSEO=user sense value
The value set in this field is sent to the logical unit as part of a negative response or as part of a Logical Unit Status (LUS) command. It may be used to inform the logical unit that the exception condition is not being raised because of a SNA-defined error as defined by IBM in the SSENSEO and SSENSMO fields (see Appendix C) but because of an application-program-related error. Alternatively, the USENSEO field in the application program may be used to further modify the information supplied in the SSENSEO and SSENSMO fields for SNA-defined errors. The user sense value is coded as any 2-byte decimal quantity or as any

2-byte framed hexadecimal or character constant (such as USENSEO=X'4F4F' or USENSEO=C'ZZ'). Register notation can also be used.

Note: *When the RPL is assembled or generated, and each time the RPL is reset to its inactive state (that is, after each synchronous request OPTCD=SYN or CHECK macro instruction), the USENSEO field is set to 0.*

Example

```
SEND1    SEND    RPL=RPL1,STYPE=REQ,CONTROL=DATA,
                 AREA=OUTBUF,RECLEN=60,CHAIN=ONLY,
                 RESPOND=(EX,FME,NRRN)
```

SEND1 sends a 60-byte data message to the logical unit identified in RPL1's ARG field. SEND1 is completed as soon as VTAM has scheduled the output operation and OUTBUF (and RPL1) are available for reuse. The RESPOND field indicates that only negative response type 1 should be returned; that is, if the message is processed normally, no response is to be returned. A RECEIVE macro (or RESP exit routine) is required to obtain the negative response, if one is returned.

Return of Status Information

After the SEND operation is completed, the following RPL fields are set:

The sequence number is placed in the SEQNO field.

The USER field contains the value that was set in the USERFLD field of the NIB when the logical unit was connected.

If POST=RESP and a negative response has been returned, the SSENSEI field may contain a system sense error code. The values that can be set in the SSENSEI field are the same as those that can be set in the SSENSEO field–namely, CPM, STATE, FI, or RR; in addition, PATH can be set when an unrecoverable path error has occurred. There is more information about these codes near the end of Appendix C.

If POST=RESP and a negative response has been returned, the SSENSMI field may contain a system sense modifier value. This field is tested as a quantity 1 byte in length. There is more information about the SSENSMI codes near the end of Appendix C.

If POST=RESP and a negative response has been returned, the USENSEI field may contain a user sense value. This field is tested as a quantity 2 bytes in length.

The value 34 (decimal) is set in the REQ field, indicating a SEND request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

In addition to the above fields, the following fields may be set when a response has been received (POST=RESP):

The CHNGDIR field indicates whether a Change Direction Command indicator or Change Direction Request indicator (non-SNA) is part of the response.

The RTYPE field indicates the type of the response.

The CHAIN field indicates the position in a chain.

The RESPOND field indicates the type of response that has been returned. This field is set in exactly the same manner as indicated above for sending a response.

Registers 0 and 15 are also set as indicated in Appendix C.

SENDCMD–Send a Network Operator Command to VTAM

It is possible for an application program to send network operator commands to VTAM and to reply to messages sent to the application program by VTAM. The SENDCMD macro instruction permits an authorized application program to send the following commands: the VARY, DISPLAY, MODIFY, and REPLY commands.

Note: *The HALT and START commands may not be issued using the SENDCMD macro instruction. They can onlv be issued by the operator at the system console.*

The SENDCMD macro instruction points to an RPL whose AREA field points to the location in the application program that contains a header followed by the command to be sent. The general format of the header and command is:

Header	Network Operator Command
←4 Bytes→	←n Bytes→

For information on using the data area and writing an application program that can issue network operator commands and receive VTAM messages, see the publication *Supplement to the VTAM Macro Language Guide for the Program Operator*, GC27-0036.

The use of the SENDCMD macro instruction must be authorized by the installation.

Name	*Operation*	*Operands*
[symbol]	**SENDCMD**	**RPL=rpl address** **[,rpl field name=new value]** . . .

RPL=rpl address
Indicates the location of the RPL that describes the SENDCMD operation.

rpl field name=new value
Indicates an RPL field name to be modified and the new value that is to be contained or represented within it. To avoid the possibility of program reassembly following some future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the SENDCMD macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction.

The following RPL operands apply to the SENDCMD macro instruction:

ACB=acb address
Indicates the ACB that identifies the application program.

AREA=command address
Indicates the address of the header and command to be sent to VTAM.

RECLEN=command length
Indicates the number of bytes to be sent to VTAM. The number specified must be equal to the length of the header and command pointed to by the AREA field. If the RECLEN field is set to 0, the AREA field is not examined. The maximum length is 130.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) SENDCMD operation is completed. If EXIT is specified, the RPL exit routine is scheduled. Otherwise, the ECB is posted, and CHECK or WAIT must be used to determine when posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM also uses the ECB-EXIT field as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
SYN specifies that control is returned to the application program when the SENDCMD operation is completed. ASY specifies that control is returned as soon as VTAM has accepted the SENDCMD request. Once the operation has been completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB-EXIT field. See the RPL macro instruction for more information.

Example

```
SENDCMD1  SENDCMD   RPL=RPL1,AREA=CMDBUF,RECLEN=CMDLEN
          .
          .
          .
CMDBUF    DS        10F               COMMAND BUFFER
VARYCMD1  DC        X‘‘00’            HEADER
          DC        X‘‘03’            STATUS FIELD
          DC        X‘‘0001’          IDENTIFICATION NUMBER
          DC        CL19‘‘VARY NET,ACT,ID=LU1’      COMMAND
CMDEND    EQU       *
CMDLEN    EQU       CMDEND-VARYCMD1
```

SENDCMD1 sends a 23-byte header and command to VTAM instructing it to activate logical unit LU1. The status field indicates that a reply is to be returned to the application program.

Return of Status Information

After the SENDCMD operation is completed, the following RPL fields are set:

The value 39 (decimal) is set in the REQ field, indicating a SENDCMD request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indiciated in Appendix C.

SESSIONC–Send an SDT, Clear, or STSN Command to a Logical Unit (Record Mode Only)

SESSIONC sends Start Data Traffic (SDT), Clear, and Set and Test Sequence Number (STSN) commands to a specific logical unit (Figure 13). The use of these commands is determined by the transmission services profile that is in use. See Appendix J for a description of the profiles and the commands that can be used.

The effect of an SDT command is to permit the application program to send to the logical unit and to permit the logical unit to send to the application program.

SESSIONC is used in one of three major ways:

Send a Start Data Traffic command to a logical unit.

1 CONTROL=SDT

Allows the flow of messages and responses to begin (or resume).

Send a Clear command to a logical unit.

2 CONTROL=CLEAR

Prevents the flow of messages and responses (but not other SESSIONC commands). Neither the application program nor the logical unit can send. All pending SEND, RECEIVE, and RESETSR macro instructions are cancelled with a physical error return code. Sequence numbers are reset to 0.

Send a Set and Test Sequence Number command to a logical unit.

3 CONTROL=STSN

Setting of IBSQAC or OBSQAC Field	Action on Associated Sequence Number
SET	Reset the sequence number and also send it to the logical unit.
TESTSET	Reset the sequence number, send it to the logical unit, and obtain a reply.
INVALID	Obtain the sequence number from the logical unit.
IGNORE	Do not reset the sequence number, ignore this part of the STSN command.

Figure 13. The Major SESSIONC Options

The effect of a Clear command is to immediately stop the flow of all messages and responses between the logical unit and the application program and to reset sequence numbers to 0. All pending I/O requests for the logical unit are canceled and all incoming and outgoing messages and responses that have not yet been received are discarded.

When the application program and the logical unit discover that their inbound and outbound sequence are different, the application program uses STSN commands to communicate with the logical unit. The purpose is to establish the correct sequence numbers while traffic flow is suspended. STSN commands are used in conjunction with SDT and Clear commands as described in *VTAM Macro Language Guide*.

There are four STSN commands that the application program can send to the logical unit: SET, TESTSET, INVALID, and IGNORE. The effects of these STSN commands are discussed below under the IBSQAC and OBSQAC operand descriptions (see Figure 14).

Value of the IBSQAC or OBSQAC field when SESSIONC was issued		**Possible IBSQAC or OBSQAC field setting when SESSIONC was completed without error. (Value refers to corresponding IBSQVAL or OBSQVAL field setting.)**	
SET	Sequence number set to value in IBSQVAL or OBSQVAL field.	TESTPOS	Logical unit agrees with value. No value is returned.
		RESET	Logical unit requests another STSN command. No value returned. (VTAM and certain logical units accept this setting but it does not correspond to SNA requirements.)
TESTSET	Sequence number set to value in IBSQVAL or OBSQVAL field.	TESTPOS	Logical unit agrees with value. Value returned in IBSQVAL or OBSQVAL field.
		TESTNEG	Logical unit disagrees with value. Terminal's version returned in INSQVAL or OBSQVAL field.
		INVALID	Logical unit does not know the value. No value returned.
		RESET	Logical unit requests another STSN command. No value returned.
INVALID	Application program does not know the sequence number value.	TESTNEG	Logical unit knows the value. Value returned in IBSQVAL or OBSQVAL field.
		INVALID	Logical unit doesn't know the value either. No value returned.
		RESET	Logical unit requests another STSN command. No value returned.
IGNORE	Application program is not setting the sequence number for the associated flow. The sequence number is not changed.	TESTPOS	Logical unit ignores this part of the STSN command. No value is returned.

Figure 14. Types of STSN Commands and Their Possible Responses

A SESSIONC macro instruction can be used to send STSN commands that apply to either the inbound or the outbound sequence numbers, or that apply to both independently.

When SDT, Clear, and STSN commands are sent to the logical unit, a response type 1 is returned as part of the SESSIONC operation. That is, the command is sent as though POST=RESP and RESPOND=(NEX,FME,NRRN) had been specified on a SEND macro instruction. If a negative response is returned, the SSENSEI, SSENSMI, and USENSEI fields are set as they would be for any other type of command.

Name	*Operation*	*Operands*
[symbol]	**SESSIONC**	**RPL=rpl address** **[,rpl field name=new value]** . . .

RPL=rpl address

Indicates the location of the RPL that describes the SESSIONC operation.

rpl field name=new value

Indicates an RPL field to be modified and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the SESSIONC macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. ARG can also be coded. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register. Register notation must be used if ARG is used.

The following RPL operands apply to a SESSIONC macro instruction:

ACB=acb address

Indicates the ACB that identifies the application program and was used when the terminal was connected.

ARG=(register)

The SESSIONC macro instruction is always directed at one specific logical unit. The ARG operand specifies the register containing the CID of that logical unit. If the ARG field is not modified, the CID already in the RPL's ARG field is used.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address

Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) SESSIONC request is completed. A SESSIONC request is completed when it has been sent to the logical unit and a response to it has been returned and posted in the RPL (similar to a SEND request with POST=RESP). If EXIT is specified, the RPL exit routine is scheduled. Otherwise, the ECB is posted and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When SYN is set, control is returned to the application program when the SESSIONC request is completed (the request is completed when the command has been sent and a response has been returned). When ASY is set, control is returned as soon as VTAM has accepted the SESSIONC request; once the requested operation has been completed, the ECB is posted or the RPL exit routine is scheduled as indicated by the ECB-EXIT field. See the RPL macro instruction for more information.

CONTROL=SDT | CLEAR | STSN

CONTROL=SDT
Sends a Start Data Traffic command to the logical unit. The effect of this command is to allow the flow of messages and responses to begin (or to resume, if a Clear command has been issued to stop the flow). When SDT=SYSTEM is coded as part of the logical unit's NIB, VTAM automatically sends a Start Data Traffic command as part of the connection process. If SDT=APPL is coded instead, it is the application program's responsibility to send the command when data traffic is to begin.

CONTROL=CLEAR
Sends a Clear command to the logical unit. The effect of this command is to stop the flow of messages and responses and to discard data that is still in the network. All SEND, RECEIVE, RESETSR, and SESSIONC requests in progress are completed normally or with RTNCD=12 and FDBK2=12 (SYNAD exit routine is entered). All subsequent SEND, RECEIVE, and RESETSR requests are rejected with RTNCD=20 and FDBK2=65 (LERAD exit routine is entered). Before SESSIONC is completed, VTAM sets the inbound and outbound sequence numbers to 0.

CONTROL=STSN
Sends a Set and Test Sequence Number command to the logical unit. The effect of the STSN command depends on its type (as specified in the IBSQAC and OBSQAC fields) and the sequence number sent with it (as specified in the IBSQVAL and OBSQVAL fields).

STYPE=REQ | RESP
This field designates the type of output to be sent to a logical unit. The application program uses STYPE=REQ to request that a message be sent. STYPE=RESP is used when a response is to be sent. STYPE=REQ must be set when a SESSIONC macro instruction is issued.

BRANCH=YES | NO
If SESSIONC is being issued in an application program that is running in the privileged state under a TCB (OS/VS2 MVS only), BRANCH can be set to YES. See the RPL macro instruction for more information.

IBSQVAL=inbound sequence number
Indicates a value that is 1 less than the new value that VTAM is to begin assigning to inbound messages. The application program sets this field only if SET or TESTSET is also specified in the IBSQAC field. The IBSQVAL field may be modified by the response to the STSN command.

OBSQVAL=outbound sequence number
Indicates a value that is 1 less than the new value that VTAM is to begin assigning to outbound messages. The application program sets this field only if SET or TESTSET is also specified in the OBSQAC field. The OBSQVAL field may be modified by the response to the STSN command.

IBSQAC=SET|TESTSET|INVALID|IGNORE
OBSQAC=SET|TESTSET|INVALID|IGNORE
The IBSQAC (inbound sequence number action code) and the OBSQAC (outbound sequence number action code) fields designate the type of STSN command sent to the logical unit. The application program can set either or both of these fields. The effect of setting one is identical to the effect of setting the other, except that one applies to incoming messages and the other to outgoing messages. Figure 13 summarizes the STSN command types and the responses they can elicit from the logical unit.

IBSQAC=SET
OBSQAC=SET
Sets the inbound or outbound sequence number to the value specified in the IBSQVAL or OBSQVAL field. When SESSIONC is completed, the IBSQAC or OBSQAC field contains the logical unit's response to the new value: TESTPOS (agree) or RESET (set the sequence number again).

IBSQAC=TESTSET
OBSQAC=TESTSET
Sets the inbound or outbound sequence number as does SET, but a wider range of responses to the new value are possible: TESTPOS (agree), TESTNEG (disagree), INVALID (don't know) or RESET (set the sequence number again).

IBSQAC=INVALID
OBSQAC=INVALID
Is used to obtain the logical unit's version of the appropriate sequence number. Unlike SET and TESTSET, INVALID does not set the sequence number (INVALID is used when the application program has lost its version of the sequence number). The logical unit can reply to this type of STSN command in three ways: TESTNEG (my version enclosed), INVALID (don't know either), or RESET (set the sequence number).

IBSQAC=IGNORE
OBSQAC=IGNORE
Is used to send a sequence number to the logical unit without setting the sequence number. The logical unit does not return any action code.

Example

```
SESSC1      SESSIONC    RPL=RPL1,CONTROL=STSN,OBSQAC=TESTSET,
                        OBSQVAL=(3),IBSQAC=IGNORE
```

SESSC1 sends an STSN command to a logical unit and sets the VTAM-supplied outbound sequence number to the value contained in register 3. The logical unit, noting that the type of STSN command is TESTSET, can indicate TESTPOS, TESTNEG, INVALID, or RESET with its response. The response information is available in RPL1 when SESSC1 is completed. If OBSQAC is found by the application program to be set to TESTPOS or TESTNEG, the OBSQVAL field contains the logical unit's version of the outbound sequence number.

Return of Status Information

After the SESSIONC operation is completed, the following RPL fields are set:

The value 37 (decimal) is set in the REQ field, indicating a SESSIONC request.

The value originally set in the USERFLD field of the NIB is set in the USER field of the RPL.

The IBSQAC and/or OBSQAC fields are set to TESTPOS, TESTNEG, INVALID, or RESET depending on the codes initially set in these fields when SESSIONC was issued. Figure 13 lists the codes that can be returned for each code initially set.

The IBSQVAL and/or OBSQVAL fields contain a sequence number when the IBSQAC and/or OBSQAC field is set to TESTPOS or TESTNEG. See Figure 13.

If an exception response is returned, the SSENSEI field may contain a system sense code. The possible codes (PATH, CPM, STATE, FI, and RR) are described near the end of Appendix C.

If an exception response is returned, the SSENSMI field may contain a system sense modifier value. This value, combined with the system sense code contained in the SSENSEI field, describes the specific type of error that caused the exception condition. See Appendix C. This value is tested as a 1-byte quantity.

If an exception response is returned, the USENSEI field may contain a user sense value. This value is tested as a 2-byte quantity.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

SETLOGON–Reset an ACB's Logon Status

There are three types of SETLOGON requests: QUIESCE, START, and STOP. The QUIESCE-START-STOP option code in SETLOGON's RPL determines which type is used. None of these three versions has any effect unless the ACB was opened with MACRF=LOGON set. The MACRF=LOGON operand indicates to VTAM that the application program can accept logons.

The START version of SETLOGON causes any application program issuing INQUIRE (OPTCD=APPSTAT) to receive a return code indicating that your application program is accepting logons. The *first* SETLOGON (OPTCD=START) issued after OPEN causes VTAM to begin scheduling the LOGON exit routine for all automatic logons, for all new logons, and for any logons already queued. SETLOGON (OPTCD=START) reverses the effect of SETLOGON (OPTCD=STOP), but it does not reverse the effect of SETLOGON (OPTCD=QUIESCE).

The STOP version of SETLOGON does not close the logon queue; any CLSDST-initiated logons from other application programs or logons originating from terminals cause the LOGON exit routine to be scheduled. However, any application program issuing INQUIRE (OPTCD=APPSTAT) for your ACB receives a return code indicating that logons should not be directed at the ACB.

The QUIESCE version of SETLOGON causes VTAM to prevent queuing of logons and the scheduling of the LOGON exit routine. There is no way to reopen the logon queue short of closing the ACB and then reopening it. An application program might want to use this type of SETLOGON at the end of a day's work, prior to closing the ACB; this would give the application program a chance to handle its current load of connected logical units without being connected to new ones. Any application program issuing INQUIRE (OPTCD=APPSTAT) for your ACB will receive a return code indicating that your application program is shutting down and cannot receive logons.

To summarize:

Request		Result
OPEN	ACB's MACRF field set to NLOGON	No logon of any kind can cause the LOGON exit routine to be scheduled. SETLOGON cannot be used to permit LOGON exit routine scheduling; only closing the ACB and reopening it with MACRF=LOGON will permit this.
OPEN	ACB's MACRF field set to LOGON	LOGON exit routine scheduling can be started by a subsequent SETLOGON (OPTCD=START).
SETLOGON	RPL=RPL1, OPTCD=START	LOGON exit routine scheduling begins for all queued, new, and automatic logons. If a LOGON exit routine is available, each logon causes it to be scheduled. If a routine is not available, the logon is queued awaiting an OPNDST ACCEPT macro instruction.

Request		Result
SETLOGON	RPL=RPL1, OPTCD=STOP	Does not stop the scheduling of the LOGON exit routine, but causes application programs issuing INQUIRE (OPTCD=APPSTAT) to receive a return code indicating that logons should not be issued for your application program.
SETLOGON	RPL=RPL1, OPTCD=QUIESCE	LOGON exit routine request scheduling *permanently* closed; it can be reopened only by closing and reopening the ACB. Serves to notify other application programs issuing INQUIRE that logons cannot be accepted.

Name	*Operation*	*Operands*
[symbol]	**SETLOGON**	**RPL=rpl address** **[,rpl field name=new value] . . .**

RPL=rpl address

Indicates the location of the RPL that in turn indicates the ACB whose logon status is to be changed.

rpl field name=new value

Indicates an RPL field to be modified, and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the SETLOGON macro instruction.

Format: For *rpl field name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register.

The following RPL operands apply to a SETLOGON macro instruction:

ACB=acb address

Indicates the ACB that identifies the application program whose logon queuing status is being changed.

ECB=ecb address|INTERNAL
EXIT=rpl exit routine address

Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) SETLOGON macro instruction is completed. The macro instruction is completed immediately, subject to delays due to possible storage shortages. If EXIT is specified, the RPL exit routine is scheduled. Otherwise, the ECB is posted, and CHECK or WAIT must be used to determine when posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When SYN is set, control is returned to the application program immediately, subject to possible delays due to storage shortages. When ASY is set, control is immediately returned to the application program, regardless of possible delays in the completion of the macro instruction. When the macro instruciton is completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the ECB-EXIT field.

OPTCD=QUIESCE | START | STOP
When QUIESCE is set, no more logons can be queued for your application program. When START is used, the scheduling of the LOGON exit routine begins for all new, queued, and automatic logons. When STOP is used, users of INQUIRE (OPTCD=APPSTAT) receive a return code indicating that logons should not be directed at your application program; however, if logons are directed at your application program from any source, VTAM will accept them and schedule the logon routine.

Example

```
         OPEN      ACB1
BEGIN    SETLOGON  RPL=RPL1,ACB=ACB1,OPTCD=START
         .
         .
         .
TOOMANY  SETLOGON  RPL=RPL1,ACB=ACB1,OPTCD=STOP
         .
         .
         .
RESUME   SETLOGON  RPL=RPL1,ACB=ACB1,OPTCD=START
         .
         .
         .
NOMORE   SETLOGON  RPL=RPL1,ACB=ACB1,OPTCD=QUIESCE
         .
         .
         .
ACB1     ACB       APPLID=APPLNAME,MACRF=LOGON
APPLNAME DC        X'05'
         DC        CL5'STOCK'
```

Before BEGIN is executed, the application program's LOGON exit routine cannot be scheduled. Once BEGIN has completed however, STOCK's LOGON exit routine is scheduled as each logon occurs. (If the installation has defined a number of automatic logons, they will each cause the LOGON exit routine to be scheduled in turn.)

TOOMANY causes VTAM to flag the application program as temporarily unwilling to accept logons. It does not prevent logons from being queued for STOCK or prevent STOCK's exit routine from being scheduled. If an application program that wants to direct a logon at ACB1 first issues INQUIRE (OPTCD=APPSTAT), it will receive a return code indicating that logons should not be issued for STOCK. The IBM-supplied network solicitor program always issues this type of INQUIRE and honors the flag set by TOOMANY.

RESUME reverses the effect of TOOMANY; application programs issuing INQUIRE (OPTCD=APPSTAT) will receive a return code indicating that logons are being accepted (the same return code that results if INQUIRE is issued after BEGIN but before TOOMANY).

NOMORE closes the logons queue. An INQUIRE issued by another application program would indicate this, and any attempt to direct a logon request to STOCK would fail. If any logons are queued when QUIESCE is issued, they remain queued.

Return of Status Information

After SETLOGON processing is finished, the following RPL fields are set:

If OPTCD=QUIESCE, the number of logons queued for the ACB is set in the RECLEN field. This quantity can be examined with the SHOWCB macro instruction (a byte work area is required) or the TESTCB macro instruction.

If this is the first SETLOGON (OPTCD=START) request issued after opening the ACB, and a temporary storage shortage error occurs while attempting to schedule the LOGON exit routine, the SETLOGON will fail (RTNCD=8,FDBK2=0). The LOGON exit routine will have been successfully scheduled for all logons prior to the storage shortage. Any remaining automatic logons will not be processed by additional SETLOGON requests. The application program can request that the network operator issue a VARY command for each of the logons. This situation is caused by allocating too little storage for the user exit control blocks and may be corrected by the network operator when VTAM is started by increasing the value of the UECBUF parameter.

The value 21 (decimal) is set in the REQ field, indicating a SETLOGON request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

SHOWCB–Extract the Contents of Control Block Fields

SHOWCB extracts the contents of one or more ACB, EXLST, RPL, or NIB fields and places them into an area designated by the application program. The SHOWCB user specifies the address of a control block and the names of the fields whose contents are to be extracted. The field names are the same as the keywords of the ACB, EXLST, RPL, and NIB macro instructions. Any keyword of these macro instructions can be used as a field name in the SHOWCB macro instruction. See Appendix E for a list and explanation of the valid formats in which the SHOWCB operands can be specified.

Control block fields that can be operated on by SHOWCB are not limited, however, to fields that can be set by the application programmer in the ACB, EXLST, RPL, and NIB macros. Several additional fields whose contents are set only by VTAM can also be displayed with SHOWCB. All of the fields applicable for SHOWCB are shown in Figure 14 at the end of the SHOWCB macro instruction description.

The user of SHOWCB must use the AREA and LENGTH operands to indicate the location and length of the area where the fields will be placed. The content of each field is placed there contiguously, in the order indicated by the FIELDS operand. If the area is too short to hold all of the fields, SHOWCB does not modify the area but returns error codes in register 0 and 15. Figure 14 shows the required lengths for all the control block fields that can be displayed with SHOWCB.

List, generate, and execute forms of the SHOWCB macro instruction are available; they are designated by the MF operand.

Name	*Operation*	*Operands*
[symbol]	**SHOWCB**	**AM=VTAM** **[{ , ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address }]** **, FIELDS=field name |(field name,...)** **, AREA=data area address** **, LENGTH=data area length** **[, MF=list, generate, or execute form parameters]**

AM=VTAM

Identifies this macro instruction as a macro instruction capable of manipulating a VTAM control block. This operand is required.

ACB=acb address
EXLST=exit list address
RPL=rpl address
NIB=nib address

Indicates the type and location of the control block whose fields are to be extracted. One of these operands must be specified unless a control block *length* (and only the length) is being extracted. That is, if FIELDS=ACBLEN, FIELDS=EXLLEN, FIELDS=RPLLEN, or FIELDS=NIBLEN is specified, no specific control block need be specified.

FIELDS=field name|(field name,...)

Indicates the control block field or fields whose contents are to be extracted.

For *field name,* code one of the field names that appear in the first column of the table that appears at the end of this macro instruction description (Figure 14). Most of these field names correspond to keywords of the ACB, EXLST, RPL, and NIB macro instructions. Only those fields associated with *one* control block can be specified (those for the control block whose address is supplied in the first operand).

AREA=data area address

Indicates the location of the storage area in the application program where the contents of the control block field or fields are to be placed. This work area must begin on a fullword boundary.

LENGTH=data area length

Indicates the length (in bytes) of the storage area designated by the AREA operand.

If this length is insufficient, SHOWCB returns a value of 4 in register 15 (unsuccessful completion) and a value of 9 in register 0 (insufficient length). The required length for each field is shown in the second column of the table that appears at the end of this macro instruction description.

MF=list, generate, or execute form parameters

Indicates that a list, generate, or execute form of SHOWCB is to be used. Omitting this operand causes the standard form of SHOWCB to be used. See Appendix F for a description of the nonstandard forms of SHOWCB.

Examples

```
SHOW1    SHOWCB    NIB=NIB1,FIELDS=NAME,AREA=NAME1,
                   LENGTH=8,AM=VTAM
```

SHOW1 extracts the contents of NIB1's NAME field and places it in NAME1.

```
SHOW2    SHOWCB    RPL=RPL1,FIELDS=(FDBK,ARG,AREA,RECLEN),
                   AREA=(3),LENGTH=16,AM=VTAM
```

SHOW2 extracts the contents of RPL1's FDBK, ARG, AREA, and RECLEN fields and places them (in that order) in a storage area. The address of this storage area must be in register 3 when SHOW2 is executed. Note that LENGTH indicates a storage area length great enough to accomodate all four fields.

Return of Status Information

After SHOWCB processing is completed, VTAM sets register 15 to indicate successful or unsuccessful completion. If the operation is completed successfully, register 15 is set to 0 and register 0 contains the total number of bytes that SHOWCB extracted and placed in the work area. If the operation completes unsuccessfully, register 15 is set to either 4, 8, or 12 (DOS/VS only). If it is set to 4 or 12, register 0 is also set indicating the specific nature of the error (see Appendix D).

Control Block Fields Applicable for SHOWCB

The field names shown in the first column of Figure 15 are the values that can be supplied for the FIELDS operand of the SHOWCB macro instruction. The lengths shown in the second column are the number of bytes of storage that must be reserved for each field; the sum of all the fields to be displayed by SHOWCB should be the value for the LENGTH operand.

ACB Fields

Field Name	Length (bytes)	Description
APPLID	4	Address of application program's symbolic name
PASSWD	4	Address of password
EXLST	4	Address of exit list
ACBLEN	4	Length of ACB, in bytes
ERROR	4	OPEN and CLOSE completion code

EXLST Fields

Field Name	Length (bytes)	Description
LERAD	4	Address of exit-routine
SYNAD	4	
DFASY	4	
RESP	4	
SCIP	4	
TPEND	4	
RELREQ	4	
LOGON	4	
LOSTERM	4	
ATTN	4	
EXLLEN	4	Length of exit list, in bytes

RPL Fields

Field Name	Length (bytes)	Description
ACB	4	Address of ACB
NIB	4	Address of NIB
ARG	4	CID of terminal
AREA	4	Address of I/O work area
AREALEN	4	Length of AREA work area, in bytes
RECLEN	4	Length of data in AREA work area, in bytes
AAREA	4	Address of alternate I/O area
AAREALN	4	Length of AAREA, in bytes
ARECLEN	4	Length of data placed in alternate I/O area
ECB	4	ECB or address of an ECB
EXIT	4	Address of RPL exit-routine
RTNCD	4	Recovery return code (1 byte, right-adjusted)
FDBK2	4	Specific error return code (1 byte, right-adjusted)
FDBK	4	Status information about successful input operations (1 byte, right-adjusted)
USER	4	The data originally placed in a NIB's USERFLD field
REQ	4	Request-type code (1 byte, right-adjusted)
RPLLEN	4	Length of RPL, in bytes
SENSE	4	Device sense and status (2 bytes, right-adjusted)
SEQNO	4	Sequence number (2 bytes, right-adjusted)
SSENSMO	4	Outbound system sense modifier (1 byte, right-adjusted)
USENSEO	4	Outbound user sense (2 bytes, right-adjusted)
SSENSMI	4	Inbound system sense modifier (1 byte, right-adjusted)
USENSEI	4	Inbound user sense (2 bytes, right-adjusted)
IBSQVAL	4	Inbound sequence number for STSN command (2 bytes, right-adjusted)
OBSQVAL	4	Outbound sequence number for STSN command (2 bytes, right-adjusted)
SIGDATA	4	Information included with signal command

NIB Fields

Field Name	Length (bytes)	Description
NAME	8	Symbolic name of terminal
USERFLD	4	Arbitrary data associated with NAME
CID	4	Communication ID
NIBLEN	4	Length of NIB, in bytes
DEVCHAR	8	Device characteristics (see Appendix H)
EXLST	4	Address of NIB-oriented exit list
RESPLIM	4	Maximum number of concurrent SEND (POST=RESP) macros
LOGMODE	8	Logon mode
BNDAREA	4	Address of bind area

Figure 15. Control Block Fields That Can Be Extracted with SHOWCB

SIMLOGON–Generate a Simulated Logon

A logon can be initiated (1) by VTAM, in accordance with VTAM definition specifications (automatic logon); (2) by some other application program, which directs the logon toward your application program; (3) by the network operator with a VARY command; (4) by the network solicitor or by the terminal itself; or (5) by your own application program. The latter is called a *simulated logon,* and the program uses the SIMLOGON macro instruction to generate it.

By issuing SIMLOGON, the application program can use its LOGON exit routine to service simulated logons that it generated on behalf of a terminal. The effect of the simulated logon is to schedule the ACB's LOGON exit routine. It is the subsequent OPNDST (OPTCD=ACCEPT) that causes the actual connection to take place. SIMLOGON is equivalent to an OPNDST connection request with an ACQUIRE option, except that the LOGON exit routine handles the connection request.

The terminal may already be connected to or queued for connection to another application program when you issue the SIMLOGON macro instruction. If the SIMLOGON macro instruction was issued with OPTCD=(Q,RELRQ) and the terminal is in session with another application program, VTAM schedules that application program's RELREQ exit routine.

Note: *An application program should never issue a SIMLOGON macro instruction if the ACB was opened with MACRF=NLOGON.*

When a SIMLOGON (OPTCD=Q) macro instruction is issued for a dial-in non-SNA terminal, the LOGON exit routine is scheduled when the terminal dials in. For a dial-in BSC terminal, the first I/O request following connection must be a SOLICIT or READ (OPTCD=SPEC) macro instruction.

When a SIMLOGON (OPTCD=Q) macro instruction is issued for a dial-in SNA terminal, the LOGON exit routine is scheduled immediately without regard to whether the logical unit has actually dialed in. If the logical unit has already dialed in, the connection will be completed successfully; if the logical unit has not dialed in, the OPNDST will fail. To prevent the LOGON exit routine from being scheduled before the logical unit has dialed in, the SIMLOGON should be issued, with OPTCD=NQ. If the logical unit has not dialed in, the SIMLOGON operation will fail and the LOGON exit will not be scheduled.

The LOGON exit routine is scheduled for both SNA and non-SNA terminals when the terminal is available, but the terminals are actually dialed at different times. Non-SNA terminals are dialed when the first I/O request is issued for that terminal. SNA terminals are dialed when the OPNDST macro instruction is issued.

If you acquire a dial-in terminal without the ID verification feature, you can establish the identity of the terminal only by issuing I/O requests and obtaining information from the terminal operator. If the terminal has the ID verification feature, INQUIRE (OPTCD=BSCID) can be used to obtain the terminal's identification sequence.

The SIMLOGON macro instruction can optionally be used to send a logon message along with the logon. See the AREA and RECLEN operands below for details.

The use of SIMLOGON must be authorized for the application program by the user.

Name	*Operation*	*Operands*
[symbol]	**SIMLOGON**	**RPL=rpl address** **[,rpl field name=new value]** . . .

RPL=rpl address

Indicates the location of the RPL to be used during SIMLOGON processing. When SIMLOGON is executed, the NIB field of this RPL should contain the address of a NIB or list of NIBs whose associated terminals are to be considered as the sources of the logons. The ACB field of the RPL must contain the address of the ACB to which the simulated logon is to be directed.

rpl field name=new value

Indicates an RPL field to be modified and the new value that is to be contained within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the SIMLOGON macro instruction.

Format: For *rpl name* code the keyword of the RPL macro instruction operand that corresponds to the RPL field to be modified. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register.

The following RPL operands apply to a SIMLOGON macro instruction:

ACB=acb address

Indicates the ACB that identifies the application program to which the simulated logon is to be directed.

NIB=nib address

Indicates the NIB whose NAME field identifies the terminal for which the simulated logon is to be generated and whose LOGMODE field specifies the logon mode name to be used. If the NIB field contains the address of a list of NIBs, logons will be generated on behalf of all the terminals of that list.

AREA=logon message address

Indicates the location of the data that VTAM is to pass to the application program as a logon message. The content and format of the data is determined by the sending and receiving application programs. The logon message is equivalent to the user data portion of an Initiate Self or a character coded logon. The other application program issues INQUIRE (OPTCD=LOGONMSG or SESSPARM) to get this data.

RECLEN=logon message length
Indicates how many bytes of data are to be passed as the logon message. If no logon message is to be sent, RECLEN should be set to 0.

ECB=ecb address| INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) SIMLOGON operation is completed. The macro instruction is completed immediately, subject to delays due to possible storage shortages. If the Q option (described below) is used, the completion of the *operation* (that is, the generation of the logon) may occur at a much later time. When the LOGON exit routine is scheduled, the SIMLOGON is posted complete. If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN|ASY
When the SYN option code is set, control is returned to the application program when the macro instruction has been completed. When ASY is set, control is returned as soon as VTAM has accepted the request. Once the operation has been completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the setting of the ECB-EXIT field.

OPTCD=CONANY|CONALL
When CONANY is set, a logon is generated for the first available terminal in the NIB list. If the Q option is set, a logon will be generated when the first terminal becomes available. If NQ is set and a terminal is available, the logon is generated immediately; however, if no terminals are available, a logon is not generated.

When CONALL is set, a logon is generated for each available terminal in the NIB list. If Q is set, logons will be generated as each terminal becomes available. If NQ is set and all the terminals are available, the logons will be generated immediately; however, if all the terminals are not available, no logons are generated.

If there is only one NIB, the setting of this option code does not matter. See the *Macro Language Guide* for more information on specific uses of CONANY and CONALL with OPNDST and SIMLOGON.

OPTCD=Q| NQ
When Q is set, the LOGON exit routine is scheduled as the terminal or terminals become available. When NQ is set, the SIMLOGON operation fails if the terminal or terminals are not immediately available.

OPTCD=RELRQ| NRELRQ
This option code is meaningful only if the Q option code is set and if the terminal (or terminals) for which the logon is to be generated, is already connected to another application program. When RELRQ is set, VTAM invokes the owning application program's RELREQ exit routine. If NRELRQ is set, the owning

application program is not notified of your request for its terminal. (The owning application program is the application program to which the terminal is currently connected.)

Example

```
SIM1       SIMLOGON   RPL=RPL1,ACB=ACB1,NIB=NIBLIST1,
                      AREA=LGNMSG,RECLEN=60,EXIT=CHECKPGM,
                      OPTCD=(ASY,CONALL,NRELRQ,Q)
           .
           .
           .
LGNMSG     DC         CL60'LOGON FROM NIBLIST1 STATION'
ACB1       ACB        MACRF=LOGON
NIBLIST1   NIB        NAME=STATIONA,MODE=RECORD,LISTEND=NO,
                      LOGMODE=BATCH
           NIB        NAME=STATIONB,MODE=RECORD,LISTEND=NO
           NIB        NAME=STATIONC,MODE=RECORD,LISTEND=YES
```

SIM1 generates simulated logons for ACB1 from all of the logical units represented in NIBLIST1. Each logon will be accompanied by a 60-byte logon message taken from LGNMSG. The logon mode for STATIONA is BATCH; STATIONB and STATIONC will use a default logon mode. The CONALL option code indicates that requests are to be generated for all the logical units represented in the list. NRELRQ indicates that if any of the logical units of NIBLIST1 are connected to an ACB other than ACB1, that ACB's RELREQ exit routine is not to be scheduled. After the SIM1 operation is completed, control is transferred to CHECKPGM.

Return of Status Information

When the SIMLOGON operation is completed; the following RPL fields are set:

The value 22 (decimal) is placed in the REQ field, indicating a SIMLOGON request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

SOLICIT—Obtain Data from a Terminal (Basic Mode Only)

The SOLICIT macro instruction obtains data from one or more connected BSC, start-stop, or local non-SNA terminals. A subsequent READ macro instruction is required to move the data from VTAM buffers to the input area provided by the application program.

SOLICIT performs the preparation or polling required to obtain the data and supplies appropriate line-control responses as blocks of data are obtained. (SOLICIT does not unlock the keyboard of a 3270 display station; this can be done with a WRITE macro instruction if an unlock-keyboard control character is included in the data stream.)

The SOLICIT macro instruction is completed as soon as VTAM has accepted the request. The actual solicitation of data continues as indicated by the BLOCK-MSG-TRANS-CONT processing option used when the terminal being solicited was connected. The effect of these options is summarized below; see the NIB macro instruction description (including Figure 6) for more information.

PROC=BLOCK: One block of data ending in an EOB line control character (for start-stop devices) or an ETB line-control character (for binary synchronous devices) is obtained.

PROC=MSG: Blocks of data are continuously obtained until a block containing an EOT character (for start-stop devices) or an ETX character (for binary synchronous devices) is received. In effect, this means that data is solicited from the terminal until an entire message has been received.

PROC=TRANS: Blocks of data are continuously obtained until a block containing an EOT character is recognized. In effect, this means that data is solicited from the terminal until an entire transmission has been received.

PROC=CONT: Blocks of data are continuously solicited from the terminal. This solicitation continues indefinitely, unless the request is canceled with the RESET macro instruction, or the terminal becomes disconnected from the program.

SOLICIT does not obtain data from a terminal if the application program has not read all previously solicited data, or if the CS option code was in effect for the last I/O request directed at the terminal.

Any I/O errors that occur during solicit operations for a given terminal become known to the program only when it next issues a READ or WRITE macro for that terminal.

Name	*Operation*	*Operands*
[symbol]	**SOLICIT**	**RPL=rpl address** **[, rpl field name=new value]...**

RPL=rpl address

Indicates the location of the RPL that governs the solicit operation.

rpl field name=new value

Indicates an RPL field to be modified and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the SOLICIT macro instruction.

Format: For *rpl field name,* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. ARG can also be coded. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register. The value supplied for the ARG keyword must indicate a register.

The following RPL operands apply to a SOLICIT macro instruction:

ACB=acb address
Indicates the ACB that identifies the application program.

ARG=(register)
If a specific terminal is to be solicited, the ARG field of the RPL must contain the CID of that terminal. ARG=(register) is indicated here because register notation must be used to place the CID in the RPL with this SOLICIT macro instruction. ARG does not apply to SOLICIT when the ANY option code is set.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address
Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) SOLICIT macro instruction is completed. The macro instruction is completed immediately, subject to delays due to possible storage shortages. If EXIT, LEVENT, or GEVENT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY
When the SYN option code is set, control is returned to the application program when the SOLICIT macro instruction has been completed. When ASY is set, control is returned when the macro instruction has been accepted. Although a SOLICIT macro instruction is usually completed immediately after the macro instruction has been accepted, a storage shortage could cause a delay. Upon return of control from an asynchronous SOLICIT, the ECB is posted or or the RPL exit routine is scheduled, as indicated by the setting of the ECB-EXIT field.

OPTCD=CA | CS
When the CA option code is set, the data obtained from the solicit operation is available for a subsequent READ (OPTCD=ANY). When CS is set instead, and the

SPEC option code is also set, only a subsequent READ (OPTCD=SPEC) can be used to retrieve the data obtained by the solicit operation. If the ANY option code is set, the CA-CS option code is treated as though CA had been specified, regardless of the actual setting.

OPTCD=SPEC | ANY
When the SPEC option code is set, data is solicited from only one terminal; namely the terminal whose CID has been placed in the ARG field of the SOLICIT macro's RPL. When ANY is set, data is solicited from all terminals that are connected to the program and are not already being solicited. If only one terminal is available for solicitation, avoid using SOLICIT (OPTCD=ANY). It will work, but it takes more time than SOLICIT (OPTCD=SPEC).

Examples

```
SLCT1      SOLICIT    RPL=RPL1,OPTCD=ANY
           .
           .
           .
RPL1       RPL        ACB=ACB1
```

SLCT1 causes data to be solicited from any terminal that has been connected through ACB1 and is not currently engaged in communication with the application program.

```
SLCT2      SOLICIT    RPL=RPL2,OPTCD=SPEC,ARG=(6)
           .
           .
           .
RPL2       RPL        ACB=ACB1
```

SLCT2, which represents a more likely use of SOLICIT, causes data to be solicited from the terminal whose CID is in RPL2's ARG field.

Return of Status Information

Control is returned to the program when VTAM has accepted the request, not when the actual I/O activity is eventually completed. After control has been returned, these RPL fields are set:

If OPTCD=SPEC is specified for in SOLICIT, the USER field is set. When a NIB is created, the application program has the option of specifying any value in the USERFLD field of that NIB. When the SOLICIT macro instruction is subsequently issued for the terminal connected with that NIB, VTAM obtains the value that was set in the USERFLD field and places it in the RPL's USER field.

The value 30 (decimal) is set in the REQ field, indicating a SOLICIT request.

The RTNCD and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

TESTCB–Test the Contents of a Control Block Field

TESTCB compares the contents of a specified ACB, RPL, EXLST, or NIB field with a value supplied with the macro instruction, and sets the PSW condition code accordingly.

The user of the TESTCB macro instruction indicates a particular control block, identifies a single field within that control block, and supplies the value against which the contents of that field are to be tested. Figure 15 lists the control block fields that can be tested.

The operands for testing control block fields are used in much the same way as operands for modifying or setting control block fields in macros like MODCB or GENCB. For example, RECLEN=200 in a MODCB macro places the value 200 in the RECLEN field of an RPL; if RECLEN=200 is specified in a TESTCB macro instruction, the contents of the RECLEN field are compared with the value 200. See Appendix E for a list and explanation of the various formats in which the TESTCB operands can be coded.

The test performed by TESTCB is a logical comparison between the field's actual content and the specified value. The condition code indicates a high, equal, or low result (with the actual content considered as the "A" comparand of the "A:B" comparison). The TESTCB macro instruction can be followed by any branching instructions that are valid following a compare instruction.

TESTCB can be used to test any control block field whose content can be set by the application program, as well as some of the control block fields whose contents are set by VTAM. The explanation below of the *field name* operand indicates the fields that can be tested.

With the ERET operand of the TESTCB macro instruction, the application program can supply the address of an error-handling routine. This routine is invoked if some error condition prevents the test from being performed correctly.

List, generate, and execute forms of TESTCB are available; they are designated by the MF operand.

Name	*Operation*	*Operands*
[symbol]	**TESTCB**	**AM=VTAM** **[{ , ACB=acb address , EXLST=exit list address , RPL=rpl address , NIB=nib address }]** **, field name=test value** **[, ERET=error exit routine address]** **[, MF=list, generate, or execute form parameters]**

AM=VTAM
Identifies this macro instruction as a VTAM macro instruction. This operand is required.

ACB=acb address
EXLST=exit list address
RPL=rpl address
NIB=nib address

Indicates the type and location of the control block whose field is to be tested.

This operand is normally required, but can be omitted if a control block *length* is being tested. (Control block lengths are tested by specifying ACBLEN, EXLLEN, RPLLEN, or NIBLEN for the TESTCB macro instruction). Since every control block of a given type is the same length for a given operating system, it is not necessary for you to indicate *which* ACB, EXLST, RPL, or NIB you want to know the length of.

field name=test value

Indicates a control block field and a value that its contents are to be tested against. For *field name,* code one of the field names that appear in the table at the end of this macro instruction description (Figure 15).

The rules for coding *test value* are defined and summarized in Appendix E.

Examples:

```
TESTCB    ACB=ACB1,PASSWD=(6),AM=VTAM
TESTCB    EXLST=EXLST1,SYNAD=SYNADPGM,AM=VTAM
TESTCB    RPL=RPL1,AREALEN=64,AM=VTAM
TESTCB    NIB=NIB1,LISTEND=YES,AM=VTAM

TESTCB    ACB=ACB1,OFLAGS=OPEN,AM=VTAM
TESTCB    RPL=RPL1,RPLLEN=38,AM=VTAM
```

RPL option codes or NIB processing options (including combinations of them) can also be tested. The test results in an equal condition code if *all* of the specified options are present. The first example below shows how to test for the presence of the SPEC, CS, *and* BLK option codes of an RPL. The second example illustrates how to code a similar test for the MSG, CONFTXT, *and* MONITOR processing options of a NIB.

Examples:

```
TESTCB    RPL=RPL1,OPTCD=(SPEC,CS,BLK),AM=VTAM
TESTCB    NIB=NIB1,PROC=(MSG,CONFTX,MONITOR),
          AM=VTAM
```

ERET=error exit routine address

Indicates the location of a routine to be entered if TESTCB processing encounters a situation that prevents it from performing the test.

When the ERET routine receives control, register 15 contains a return code. If this return code indicates an error, register 0 will contain an error return code that indicates the nature of the error. These return codes are described in Appendix D (and are summarized below). Register 14 contains the address of the ERET exit routine and the remaining registers are unchanged.

Note: *If this operand is omitted, the program instructions that follow the TESTCB macro instruction should check register 15 to determine whether an error occurred (indicating that the PSW condition code is meaningless) or not. To make this check without disturbing the condition code, a branching table based on register 15 can be used.*

MF=list, generate, or execute form parameters

Indicates that a list, generate, or execute form of TESTCB is to be used. Omitting this operand causes the standard form of TESTCB to be used. See Appendix F for a description of the nonstandard forms of TESTCB.

Return of Status Information

After TESTCB processing is finished and control is either passed to the ERET error routine or returned to the next sequential instruction, register 15 indicates whether or not the test was completed successfully. If the test completed successfully, register 15 is set to 0; if it completed unsuccessfully, register 15 is set to either 4, 8, or 12. If it is set to 4, register 0 is also set indicating the specific nature of the error (see Appendix D).

Control Block Fields Applicable for TESTCB

The field names shown in the first column of Figure 16 are the values that can be coded for the *field name* operand of the TESTCB macro instruction. The second column indicates the number of bytes that each field occupies. No lengths are shown for fields that can only be tested using fixed values (for example, MACRF=LOGON or CONTROL=QEC).

ACB Fields

Field Name	Length (bytes)	Description
APPLID	4	Address of application program's symbolic name
PASSWD	4	Address of password
EXLST	4	Address of exit list
ACBLEN	2	Length of ACB, in bytes
ERROR	1	OPEN and CLOSE completion codes
OFLAGS	-	ACB open-closed indicator (OFLAGS=OPEN)
MACRF	-	Logon request status (MACRF=LOGON|NLOGON)

EXLST Fields

Field Name	Length (bytes)	Description
LERAD	4	Address of exit-routine
SYNAD	4	
DFASY	4	
RESP	4	
SCIP	4	
TPEND	4	
RELREQ	4	
LOGON	4	
LOSTERM	4	
ATTN	4	
EXLLEN	2	Length of exit list, in bytes

RPL Fields

Field Name	Length (bytes)	Description
ACB	4	Address of ACB
NIB	4	Address of NIB
ARG	4	CID of terminal
AREA	4	Address of I/O work area
AREALEN	4	Length of AREA work area, in bytes
RECLEN	4	Length of data in AREA work area, in bytes
AAREA	4	Address of alternate I/O work area
AAREALN	4	Length of AAREA work area, in bytes
ARECLEN	4	Length of data in AAREA work area, in bytes
ECB	4	ECB or address of ECB
EXIT	4	Address of RPL exit-routine
RTNCD	1	General return code
FDBK2	1	Specific error return code
FDBK	1	Additional status information
DATAFLG	-	Alias for FDBK
IO	-	RPL internal ECB post bit on (IO=COMPLETE)
USER	4	USERFLD data
REQ	1	Request type code
RPLLEN	2	Length of RPL, in bytes
SENSE	2	Device sense and status information (BSC)
BRANCH	-	SRB indicator (BRANCH=YES|NO, OS/VS2 MVS only)
OPTCD	-	RPL option code
SEQNO	4	Sequence number
SSENSEO	-	Outbound system sense command
SSENSMO	1	Outbound system sense modifier value
USENSEO	2	Outbound user sense value
SSENSEI	-	Inbound system sense command
SSENSMI	1	Inbound system sense modifier value
USENSEI	2	Inbound user sense value
IBSQAC	-	Inbound action code for STSN command
OBSQAC	-	Outbound action code for STSN command
IBSQVAL	4	Inbound sequence number for STSN command
OBSQVAL	4	Outbound sequence number for STSN command
POST	-	Scheduled or responded output (POST=SCHED|RESP)
RESPOND	-	Response Indicator(RESPOND=[EX|NEX] [,FME|NFME] [RRN|NRRN])
CONTROL	-	Control command (CONTROL=DATA|indicator)

Figure 16 (Part 1 of 2). Control Block Fields That Can Be Tested with TESTCB

RPL Fields (Cont.)

Field Name	*Length (bytes)*	*Description*
CHAIN	-	Chain indicator (CHAIN=FIRST|MIDDLE|LAST|ONLY)
CHNGDIR	-	Change-direction indicator ([CHNGDIR=[CMD|NCMD] [,REQ|NREQ])
BRACKET	-	Bracket indicator (BRACKET=[BB|NBB] [,EB|NEB])
RTYPE	-	Receive-type indicator (DFSYN, DFASY, RESP)
STYPE	-	Send-type indicator (STYPE=REQ|RESP)
SIGDATA	4	Information included with a signal command
CODESEL	-	Type of character encoding (CODESEL=STANDARD|ALT)

NIB Fields

Field Name	*Length (bytes)*	*Description*
NAME	8	Symbolic name of terminal
USERFLD	4	Arbitrary data associated with NAME
CID	4	Communication ID
NIBLEN	2	Length of NIB, in bytes
DEVCHAR	8	Device characteristics (see Appendix H)
EXLST	4	Address of NIB-oriented exit list
RESPLIM	4	Maximum number of concurrent SEND (POST=RESP) macros
MODE	-	Mode indicator (MODE=BASIC|RECORD)
LISTEND	-	NIB list termination flag (LISTEND=YES|NO)
SDT	-	Start-data-traffic flag (SDT=APPL|SYSTEM)
PROC	-	Processing option codes
CON	-	Terminal-connected flag (CON=YES)

Figure 16 (Part 2 of 2). Control Block Fields That Can Be Tested with TESTCB

WRITE—Write a Block of Data from Program Storage to a Terminal (Basic Mode Only)

The WRITE macro instruction obtains a block of data from a designated area in application program storage and sends it to a specific start-stop or BSC terminal or local non-SNA 3270.

There are several variations for WRITE:

- The write operation can be followed automatically by a read operation, as though a READ macro instruction had been coded after the WRITE macro. This composite operation is called a *conversational* write operation.
- The write operation can be preceded by the erasure of the 2265 screen of a 2770 Data Communication Terminal or a 3270 display device.
- The unprotected portion of a display screen in a 3270 display device can be erased (with no associated write operation performed).

Note: *Following connection, you cannot write to a 3735 Programmable Buffered Terminal until you first issue a SOLICIT or READ (OPTCD=SPEC) macro instruction for the terminal.*

Should a write operation be pending (or in progress) when another WRITE macro instruction is issued, the first operation is completed before the second operation is performed. This means that several WRITE macro instructions for a particular terminal can be issued serially. If data is being solicited from a terminal when the write macro instruction is issued, the write operation is suspended until the solicitation is completed. Since this may take some time, you may wish to cancel the SOLICIT request with a RESET macro instruction. Furthermore, if the terminal is a BSC device and data is being received which is not at an ETX (message) boundary, the WRITE macro instruction fails and a return code is placed in the RPL's FDBK2 field. See the SOLICIT or READ macro instruction for a description of what constitutes and controls the completion of a solicit operation.

The TRUNC-KEEP processing option in the NIB determines how excess input data is to be handled for a conversational WRITE macro instruction. When the TRUNC processing option is in effect and the CONV option code is set, and there is too much incoming data to fit in the area indicated by AAREA, the data is truncated, the remainder is lost, and the WRITE macro instruction terminates with an I/O error. With KEEP, however, the remainder is saved and passed to the program when the next READ macro instruction is issued for that terminal.

For a READ/WRITE sequence to a 3270 device, the write will not be suspended pending completion of the solicitation of that terminal.

Name	*Operation*	*Operands*
[symbol]	**WRITE**	**RPL=rpl address** **[,rpl field name=new value]...**

RPL=rpl address

Indicates the location of the RPL that governs the write operation.

rpl field name=new value

Indicates an RPL field to be modified and the new value that is to be contained or represented within it. If you wish to avoid the possibility of program reassembly following future releases of VTAM, set the RPL field with MODCB macro instructions rather than with the WRITE macro instruction.

Format: For *rpl field name,* code the keyword of the RPL macro instruction operand that corresponds to the RPL field being modified. ARG can also be coded. The *new value* can be any value that is valid for that operand in the RPL macro instruction, or it can indicate a register. The value supplied for the ARG keyword must indicate a register.

The following RPL operands apply to the WRITE macro instruction:

ACB=acb address

Indicates the ACB that was used when the terminal was connected.

ARG=(register)

The ARG field of the RPL must contain the CID of the terminal to which the data is to be written (see the OPNDST macro instruction for an explanation of the CID). ARG=(register) is indicated here because register notation must be used when the CID is placed in the ARG field with this WRITE macro instruction.

AREA=output data address

The data contained at the location indicated by AREA is sent to the terminal. Since the application cannot determine the exact moment that VTAM moves the data from this area, the area should not be reused until the WRITE macro instruction is completed.

RECLEN=output data length

The number of bytes of data indicated in the RECLEN field is sent to the terminal. If this field is set to 0 and OPTCD=LBT (explained below), an EOT is sent to the terminal. If OPTCD=BLK or OPTCD=LBM, the line control characters shown in Appendix B are sent without data.

AAREA=input data area address

When the CONV option code is set, the data *obtained from* the terminal following the output operation is placed in the storage area indicated by the AAREA field. VTAM also places the length of this data in the ARECLEN field. The AAREA field is not used if NCONV is set.

AAREALN=input data area length

AAREALN indicates the capacity of the data area pointed to by AAREA. If the amount of incoming data exceeds the capacity of this data area, the action indicated by the TRUNC-KEEP option code is taken.

ECB=ecb address | INTERNAL
EXIT=rpl exit routine address

Indicates the action to be taken by VTAM when an asynchronous (OPTCD=ASY) WRITE macro instruction is completed. The macro instruction is completed after the data has been received and acknowledged by the terminal (or, for a conversational WRITE, as soon as the input data has been moved into the application program's storage area). If EXIT is specified, the RPL exit routine is scheduled. Otherwise the ECB is posted, and CHECK or WAIT must be used to determine when the posting occurs.

If ECB=INTERNAL is specified and *synchronous* handling (SYN option) is used, VTAM uses the ECB-EXIT field in the RPL as an *internal* ECB and clears it. If *asynchronous* handling (ASY option) is used, VTAM will also use the ECB-EXIT field in the RPL as an internal ECB, but the user must issue a CHECK macro instruction to check and clear it. Specifying ECB=INTERNAL has the effect of specifying neither ECB=*ecb address* nor EXIT=*rpl exit routine address* in the RPL macro instruction. See the RPL macro instruction for more information.

OPTCD=SYN | ASY

When the SYN option code is set, control is returned to the application program when the WRITE macro instruction has been completed. When ASY is set, control is returned as soon as VTAM has accepted the request. Once the WRITE macro instruction has been completed, the ECB is posted or the RPL exit routine is scheduled, as indicated by the setting of the ECB-EXIT field.

OPTCD=BLK | LBM | LBT

These option codes determine whether the line-control characters selected by the system for transmission with the data should mark the data as the end of a block, the end of a message, or the end of a transmission.

When the BLK option code is set, the line-control characters indicated in Appendix B under "BLK" are sent with the block of data. BLK is invalid for a 3270 terminal.

When the LBMoption code is set, the line-control characters indicated in Appendix B under "LBM" are sent with the block of data.

When the LBT option code is set, the line-control characters indicated in Appendix B under "LBT" are sent with the block of data. After the block of data is acknowledged by the terminal an EOT character is sent. A sequence of consecutive write operations will not complete until the EOT character is sent. A WRITE macro instruction with OPTCD=LBM or LBT is needed to send an EOT to the terminal and release the line.

OPTCD=CONV | NCONV

When NCONV is set, no read operation follows the write operation. When the CONV option code is in effect, an input operation is performed after the block of data has been written to the terminal. The data received in response to the write operation is placed in the area indicated by the RPL's AAREA field, and the length of that data is set in the ARECLEN field.

Should the terminal merely respond with an acknowledgement and not data, the following action is taken: An EOT is sent to the terminal and the terminal is polled (or for a point-to-point line, placed in the receive state). The input data eventually received from this polling is then placed in the AAREA and ARECLEN fields. The operation is not completed until the data is received.

When a WRITE with OPTCD=CONV is issued, a second WRITE may not be issued to the same terminal until the first output operation is completed, unless the first operation is canceled with the RESET macro instruction.

OPTCD=ERASE | EAU | NERASE
The ERASE and EAU option codes indicates that one of two special variations of WRITE are to be used; NERASE indicates that an ordinary output operation is requested.

When ERASE is used, the entire display screen of (1) a 2265 display station attached to a 2770 Data Communication System or (2) a 3270 display station, is erased before the block of data is written to the terminal. ERASE cannot be specified for conversational WRITE operations (OPTCD=CONV); use the ERASELBM or ERASELBT LDOs followed by a chained READ LDO if a combined erase-write-read operation is desired.

EAU means that the unprotected portion of a 3270 display station screen is to be erased, and its keyboard unlocked. No data is sent to the terminal. EAU cannot be specified for a conversational WRITE operation; use the EAU LDO followed by a chained READ LDO if a combined erase-read operation is desired.

OPTCD=CS | CA
When CA is set, data obtained from the terminal can satisfy a READ (OPTCD=ANY) macro instruction. When CS is set, only READ (OPTCD=SPEC) macro instructions can obtain data from the terminal. See the RPL macro instruction for more information.

Examples

```
WRITE1    WRITE    RPL=RPL1,AREA=SOURCE,RECLEN=60,
                   EXIT=WRTDONE,OPTCD=ASY
```

WRITE1 sends a 60-byte block of data from SOURCE to the terminal. This example assumes that the CID for the terminal is already in RPL1's ARG field. Control is returned to the instruction following WRITE1 as soon as the data has been written to the terminal and an acknowledgment has been received. When the operation is completed, the program is interrupted and control passed to WRTDONE.

```
          MODCB    RPL=RPL2,ARG=(3)
WRITE2    WRITE    RPL=RPL2
```

WRITE2 erases the unprotected part of a 3270 display screen. The MODCB macro instruction places the contents of register 3 (the terminal's CID) into RPL2's ARG field.

```
WRITE3    WRITE    RPL=RPL3
                   .
                   .
                   .
RPL3      RPL      AREA=OUTGOING,RECLEN=120,ARG=(3),
                   AAREA=INCOMING,AAREALN=132,
                   OPTCD=(CONV,LBT)
```

WRITE3 requests a conversational WRITE operation. 120 bytes of data from OUTGOING are sent to the terminal whose CID is in register 3. Data is then read from the terminal and placed into INCOMING. If more than 132 bytes are received, the excess will be lost. Because the LBT option code is set, the operation is not completed until the terminal responds with data.

Return of Status Information

After a WRITE macro instruction has been completed, these RPL fields are set:

If the CONV option is set, the ARECLEN field indicates the number of bytes of data obtained during the input part of the conversational operation.

When a NIB is used during connection, VTAM associates the contents of the USERFLD field with the terminal. When the WRITE macro instruction is subsequently issued for the terminal, the contents of the USERFLD field is placed in the USER field of the RPL by VTAM.

The value 17 (decimal) is set in the REQ field indicating a WRITE request.

If the CONV option code is set, the FDBK field is set: it contains information concerning the input portion of the operation. See the FDBK field description in Appendix C.

The SENSE, RTNCD, and FDBK2 fields are set as indicated in Appendix C.

Registers 0 and 15 are also set as indicated in Appendix C.

APPENDIX A. SUMMARY OF CONTROL BLOCK FIELD USAGE

After you have become familiar with the workings of the VTAM macro instructions described in this book, this appendix can be used as a quick reference. It shows the following information about all of the executable macro instructions in this book:

- The control block fields that are set by the application program when (or before) the macro instruction is issued.
- The control block fields and registers that are set by VTAM during macro instruction processing.

Note: All of the control block fields that apply to the macro instruction are shown, but remember that not all fields apply to every possible variation of a macro instruction. Refer to the macro instruction descriptions if you are in doubt.

Throughout this appendix, a pointer (→) indicates that a field contains the *address* of the given item, and an equal-sign indicates that a field contains the item itself. You will also note that a horizontal dashed line is used with each macro instruction; all information *above* this line concerns information your application program supplies to the macro instruction, and all information shown *below* this line concerns information that the macro instruction passes back to your application program.

CHANGE

→ RPL: ACB field → ACB to which terminal is connected
  NIB field → modified NIB:
    CID field = CID of terminal
    USERFLD field = new data to be returned during subsequent I/O requests
    MODE field = BASIC
    PROC field = new set of processing options
  { ECB field → fullword work area }
  { ECB field = internal ECB }
  { EXIT field → RPL exit-routine }
  OPTCD field = (SYN | ASY,CA | CS)

\- -

Registers 0 and 15 = return codes
RPL: RTNCD field = recovery action return code
  FDBK2 field = specific error return code
  REQ field = request code

CHECK

→ RPL being checked

\- -

RPL set inactive
ECB cleared
Registers 0 and 15 = request code

CLOSE

→ ACB being closed

\- -

Register 15 = return code
ACB: OFLAGS field = opened or not-opened indicator
  ERROR field = specific error return code

CLSDST	→ RPL:	ACB field → ACB { NIB field → NIB: NAME field = symbolic name of terminal to be disconnected LOGMODE = logon mode name ARG field = CID of terminal to be disconnected } AAREA field → symbolic name of receiving application program AREA field → logon message RECLEN field = length of logon message { ECB field → fullword work area ECB field = internal ECB EXIT field → RPL exit-routine } OPTCD field = (PASS|RELEASE,SYN|ASY)
	Registers 0 and 15 = return codes RPL:	 RTNCD field = recovery action return code FDBK2 field = specific error return code REQ field = request code
DO	→ RPL:	ACB field → ACB ARG field = CID of terminal { ECB field → fullword work area ECB field = internal ECB EXIT field → RPL exit-routine } OPTCD field = (SYN|ASY,CS|CA) AREA field → LDO If CMD field = COPYLBM or COPYLBT, ADDR → copy control character and sending device's CID (ARG field contains receiving device's CID) LEN = 3 If CMD field = READ or READBUF, ADDR → input data area LEN = length of data area If CMD field = ERASELBM or ERASELBT, ADDR → output data LEN = length of data If CMD field = WRITE, WRITELBM, WRITELBT, WRTHDR, WRTNRLG, or WRTPRLG, ADDR → output data LEN = length of output data
	Registers 0 and 15 = return codes RPL:	 AAREA field → last LDO used USER field → data from USERFLD field of NIB RECLEN field = length of data received FDBK field = status information (if CMD field = READ) RTNCD field = recovery action return code FDBK2 field = specific error return code REQ field = request code
EXECRPL	→ RPL:	All fields appropriate for the request type (indicated in the REQ field) are valid.
GENCB	AM = VTAM BLK operand = control block type control block field name operand = value to be set in field	

COPIES operand = number of copies desired
WAREA operand → work area where blocks will be built
LENGTH operand = length of work area
MF operand = list, generate, or execute form parameters

{ Register 0 = error return code (if register 15 indicates unsuccessful completion)
Register 0 = length of generated control blocks (if built in dynamically allocated storage obtained by VTAM and register 15 indicates successful completion) }
Register 1 → generated control blocks (if built in dynamically allocated storage obtained by VTAM and register 15 indicates successful completion)
Register 15 = general return code

INQUIRE → RPL: ACB field → ACB
{ ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine }
OPTCD = (LOGONMSG | DEVCHAR | COUNTS | TERMS | BSCID | APPSTAT | CIDXLATE | TOPLOGON | SESSPARM,SYN | ASY)
If OPTCD = LOGONMSG,
NIB field → NIB: NAME field = symbolic name of terminal
AREA field → input area for logon message
AREALEN field = length of input area
If OPTCD = DEVCHAR,
{ NIB field → NIB: NAME field = symbolic name of terminal
ARG field = CID of terminal }
AREA field → input area for characteristics
AREALEN field = 8
If OPTCD = TERMS,
NIB field → NIB: NAME field = symbolic name of terminal or group of terminals as defined by the PU, LU, TERMINAL, LINE, or CLUSTER definition macro
AREA field → work area where NIBs will be built
AREALEN field = length of work area
If OPTCD = COUNTS,
AREA field → input area for data
AREALEN field = 4
If OPTCD = APPSTAT,
NIB field → NIB: NAME field = symbolic name of application program
If OPTCD = CIDXLATE,
{ NIB field → NIB: NAME field = symbolic name of terminal
ARG field = CID to be translated }
AREA field → input area for symbolic name
AREALEN field = 8
If OPTCD = TOPLOGON,
AREA field → input area for symbolic name
AREALEN field = 8
If OPTCD = BSCID,
NIB field → NIB: NAME field = symbolic name of UTERM terminal
AREA field → work area for ID verification sequence
AREALEN field = 20
If OPTCD = SESSPARM,
NIB field → NIB: NAME field = symbolic name of terminal
LOGMODE field = 0 | C' ' | logon mode

AREA field → input area for session parameters and logon message
AREALEN field = length of input area

Registers 0 and 15 = return codes
RPL: RECLEN field = length of data received (if RTNCD = 0 and FDBK2 = 5, RECLEN = total required length)
FDBK field = status information (if OPTCD = APPSTAT)
RTNCD field = recovery action return code
FDBK2 field = specific error return code
REQ field = request code

INTRPRET → RPL: ACB field → ACB
{ NIB field → NIB: NAME field = symbolic name of terminal
ARG field = CID of terminal }
AREA field → data to be interpreted
RECLEN field = length of data to be interpreted
AAREA field → work area for interpreted data
AAREALN field = 8
{ ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine }
OPTCD field = SYN|ASY

Registers 0 and 15 = return codes
RPL: ARECLEN field = length of data received (if RTNCD = 0 and FDBK2 = 5, ARECLEN = total required length)
RTNCD field = recovery action return code
FDBK2 field = specific error return code
REQ field = request code

MODCB AM = VTAM
control block type operand → control block
control block field name operand = new value to be set
MF operand = list, generate, or execute form parameters

Register 0 = error return code (if register 15 indicates unsuccessful completion)
Register 15 = general return code

OPEN → ACB being opened

Register 15 = return code
ACB: OFLAGS field = opened or not-opened indicator
ERROR field = specific error status information

OPNDST → RPL: ACB field → opened ACB to which terminal is to be connected
{ ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine }
OPTCD field = (SPEC|ANY,SYN|ASY,CS|CA,Q|NQ,CONANY|CONALL,ACQUIRE|ACCEPT)
NIB field → NIB|NIB list:
PROC field = processing options
MODE field = BASIC or RECORD
USERFLD field = data to be returned during subsequent I/O requests

LOGMODE = 0|logon mode|C' '
BNDAREA → bind area containing session parameters|0

If OPTCD = ACQUIRE,
NAME field = symbolic name of terminal
LISTEND field = YES or NO

If OPTCD = ACCEPT and ANY,
NAME field not examined
LISTEND field = YES

If OPTCD = ACCEPT and SPEC,
NAME field = symbolic name of terminal
LISTEND field = YES

Registers 0 and 15 = return codes

RPL: ARG field = CID of connected terminal (but if CONALL in effect and more than one connected, unpredictable)
RTNCD field = recovery action return code
FDBK2 field = specific error return code
REQ field = request code
AREA field → NIB:
CID field = CID of connected terminal
CON field = YES (if terminal connected)
NAME field = symbolic name of connected terminal (when ACCEPT and ANY in effect)
DEVCHAR field = device characteristics

RCVCMD → RPL: ACB field → ACB
AREA field → input area for header and message
AREALEN field = length of input area
ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine
OPTCD field = (SYN|ASY,TRUNC, Q|NQ)

Registers 0 and 15 = return codes

RPL: RECLEN field = length of input data received
REQ field = request code
RTNCD field = recovery action return code
FDBK2 field = specific error return code

READ → RPL: ACB field → ACB
ARG field = CID of source terminal
AREA field → input data area
AREALEN field = length of input data area
ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine
OPTCD field = (SPEC|ANY,SYN|ASY,CS|CA)

Registers 0 and 15 = return codes

RPL: ARG field = CID of source terminal (if OPTCD = ANY)
RECLEN field = length of input data
USER field = data from USERFLD field in NIB
FDBK field = status information
RTNCD field = recovery action return code
FDBK2 field = specific error return code
REQ field = request code

RECEIVE	→ RPL:	ACB field → ACB to which logical unit connected ARG field = CID of logical unit AREA field → input data area AREALEN field = length of input data area BRANCH field = YES or NO { ECB field → fullword work area ECB field = internal ECB EXIT field → RPL exit-routine } OPTCD field = (SYN|ASY, CA|CS, SPEC|ANY, TRUNC|KEEP| NIBTK, Q|NQ) RTYPE field = (DFSYN|NDFSYN, DFASY|NDFASY, RESP|NRESP)
	Registers 0 and 15 = return codes RPL:	ARG field = CID of logical unit completing the RECEIVE RTYPE field = type of input received RECLEN field = length of input data SEQNO field = sequence number of input RESPOND field = (EX|NEX, FME|NFME, RRN|NRRN) USER field = data from USERFLD field of NIB REQ field = request code RTNCD field = recovery action return code FDBK2 field = specific error return code CHNGDIR field = REQ|NREQ (if RTYPE = DFASY or RESP) CMD|NCMD (if RTYPE = DFSYN) CODESEL field = STANDARD|ALT BRACKET field = (BB|NBB, EB|NEB) CHAIN field = FIRST|MIDDLE|LAST|ONLY SIGDATA field = signal value (if CONTROL = SIGNAL) CONTROL field = DATA|QEC|RELQ|QC|CANCEL|CHASE|SHUTD| LUS|SIGNAL|RTR|RSHUTD|SHUTC OPTCD = FMHDR SSENSEI field = CPM|STATE |FI|RR|PATH|0 SSENSMI field = system sense modifier value (or 0) USENSEI field = user sense value (or 0)
RESET	→ RPL:	ACB field → ACB ARG field = CID of terminal { ECB field → fullword work area ECB field = internal ECB EXIT field → RPL exit-routine } OPTCD field = (SYN| ASY, CS|CA, COND|UNCOND|LOCK)
	Registers 0 and 15 = return codes RPL:	USER field = data from USERFLD field in NIB RTNCD field = recovery action return code FDBK2 field = specific error return code REQ field = request code
RESETSR	→ RPL:	ACB field → ACB to which logical unit connected ARG field = CID of receiving logical unit { ECB field → fullword work area ECB field = internal ECB EXIT field → RPL exit-routine } BRANCH field = YES or NO RTYPE field = (DFSYN|NDFSYN, DFASY|NDFASY, RESP|NRESP) OPTCD field = (SYN|ASY, CA|CS)

Registers 0 and 15 = return codes
RPL: USER field = data from USERFLD field in NIB
RTNCD field = recovery action return code
FDBK2 field = specific error return code
REQ field = request code

SEND → RPL: ACB field → ACB to which logical unit connected
ARG field = CID of receiving logical unit
AREA field → data to be sent
RECLEN field = length of data to be sent
CHNGDIR field = REQ|NREQ (if RTYPE = DFASY or RESP)
CMD|NCMD (if RTYPE = DFSYN)
BRANCH field = YES or NO
{ ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine }
RESPOND field = (EX|NEX, FME|NFME, RRN|NRRN)
RTYPE field = (DFSYN|NDFSYN, DFASY|NDFASY, RESP|NRESP)
CONTROL field = DATA|QEC|RELQ|QC|CANCEL|CHASE|SHUTD|
BID|LUS|SIGNAL
BRACKET field = (BB|NBB, EB|NEB)
SIGDATA field = signal data
CODESEL field = STANDARD|ALT
STYPE field = REQ or RESP
If STYPE = RESP,
SEQNO field = sequence number
If STYPE = RESP and RESPOND = EX,
SSENSEO field = CPM|STAT|FI|RR
SSENSMO field = system sense modifier value
USENSEO field = user sense value
If STYPE = REQ and CONTROL = DATA,
CHAIN field = FIRST|MIDDLE|LAST|ONLY
POST field = SCHED|RESP (SCHED assumed if definite response not requested)
OPTCD field = (SYN|ASY, FMHDR|NFMHDR, CS|CA)

- - -

Registers 0 and 15 = return code
RPL: USER field = data from USERFLD field of NIB
RTNCD field = recovery action return code
FDBK2 field = specific error return code
REQ field = request code
SEQNO field = sequence number
RESPOND = unpredictable
If POST = RESP and STYPE = REQ,
CHNGDIR field = (REQ|NREQ, CMD|NCMD)
RESPOND field = (EX|NEX, FME|NFME, RRN|NRRN)
SSENSEI field = CPM|STATE|FI|RR|PATH|0
SSENSMI field = system sense modifier value (or 0)
USENSEI field = user sense value (or 0)

SENDCMD → RPL: ACB field → ACB
AREA field → header and command to be sent
RECLEN field = length of header and command to be sent
{ ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine }
OPTCD field = SYN|ASY

- - -

Registers 0 and 15 = return codes
RPL: REQ field = request code
RTNCD field = recovery action return code
FDBK2 field = specific error return code

SESSIONC → RPL: ACB field → ACB to which logical unit connected
ARG field = CID of logical unit
{ ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine }
OPTCD field = SYN|ASY
STYPE field = REQ
CONTROL field = SDT|CLEAR|STSN
If CONTROL = STSN,
IBSQVAL field = inbound sequence number
IBSQAC field = SET|TESTSET|INVALID|IGNORE
OBSQVAL field = outbound sequence number
OBSQAC field = SET|TESTSET|INVALID|IGNORE

- - - - - - - - - - - - - - - - - - - -

Registers 0 and 15 = return codes
RPL: USER field = data from USERFLD field of NIB
RTNCD field = recovery action return code
FDBK2 field = specific error return code
REQ field = request code
SSENSEI field = CPM|STATE|FI|RR|PATH|0
SSENSMI field = system sense modifier value (or 0)
USENSEI field = user sense value (or 0)
If CONTROL = STSN,
IBSQVAL field = inbound sequence number
IBSQAC field = TESTPOS|TESTNEG|RESET|INVALID
OBSQVAL field = outbound sequence number
OBSQAC field = TESTPOS|TESTNEG|RESET|INVALID

SETLOGON → RPL: ACB field → ACB whose logon queuing status is to be changed
{ ECB field → fullword work area
ECB field = internal ECB
EXIT field → RPL exit-routine }
OPTCD field = (SYN|ASY, START|STOP|QUIESCE)

- - - - - - - - - - - - - - - - - - - -

Registers 0 and 15 = return codes
RPL: RECLEN field = number of queued logon requests
(if OPTCD = QUIESCE)
REQ field = request code
RTNCD field = recovery action return code
FDBK2 field = specific error return code

SHOWCB AM = VTAM
control block type operand → control block
FIELDS = control block field to be extracted
AREA → work area where contents of blocks will be placed
LENGTH operand = length of the work area
MF operand = list, generate, or execute form parameters

- - - - - - - - - - - - - - - - - - - -

Register 0 = error return code (if register 15 indicates unsuccessful completion)
Register 0 = length of the control block field extracted (if register 15 indicates successful completion)
Register 15 = general return code

SIMLOGON → RPL: ACB field → ACB
NIB field → NIB:
NAME field = symbolic name of terminal
LISTEND field = YES or NO
LOGMODE = logon mode name

		AREA field → logon message RECLEN field = length of logon message { ECB field → fullword work area ECB field = internal ECB EXIT field → RPL exit-routine } OPTCD field = (SYN | ASY, Q | NQ, CONANY | CONALL, RELRQ | NRELRQ)
		Registers 0 and 15 = return codes RPL: RTNCD field = recovery action return code FDBK2 field = specific error return code REQ field = request code
SOLICIT	→ RPL:	ACB field → ACB ARG field = CID of source terminal (if OPTCD = SPEC) { ECB field → fullword work area ECB field = internal ECB EXIT field → RPL exit-routine } OPTCD field = (SPEC | ANY, SYN | ASY, CS | CA)
		Registers 0 and 15 = return codes RPL: USER field = data from USERFLD field of NIB RTNCD field = recovery action return code FDBK2 field = specific error return code REQ field = request code
TESTCB		AM = VTAM control block type operand → control block field name operand = test value ERET operand → error exit-routine MF operand = list, generate, or execute form parameters
		Register 0 = error return code (if register 15 indicates unsuccessful completion) Register 15 = general return code PSW condition code = test result
WRITE	→ RPL.	ACB field → ACB ARG field = CID of receiving terminal AREA field → data to be written RECLEN field = length of data to be written AAREA field → input data area AAREALN field = length of input data area { ECB field → fullword work area ECB field = internal ECB EXIT field → RPL exit-routine } OPTCD field = (SYN | ASY, CS | CA, BLK | LBM | LBT, CONV | NCONV, ERASE | EAU | NERASE)
		Registers 0 and 15 = return codes RPL: ARECLEN field = length of input data USER field = data form USERFLD field of NIB FDBK field = status information RTNCD field = recovery action return code FDBK2 field = specific error return code REQ field = request code

APPENDIX B. COMMUNICATION-CONTROL CHARACTERS RECOGNIZED OR SENT BY VTAM MACROS

This appendix is only for programmers who are concerned with communication between the application program and BSC or start-stop terminals. Communication with logical units does not involve communication-control characters.

VTAM relieves the application program of the task of inserting line-control characters into outgoing data and removing them from incoming data. The application program, however, is not completely communication-control independent. For an output operation, the BLK-LBM-LBT option for the WRITE macro instruction governs which communication-control characters are inserted in the data. For a solicit operation, the BLOCK-MSG-TRANS-CONT option identifies the communication-control character that causes solicitation to stop when that character is received. The application programmer must be aware of the effect of these options.

The first three columns in Figures B-1 and B-2 show the communication-control characters that delimit the data obtained by a solicit operation. The first column shows the delimiting character when the NIB's BLOCK-MSG-TRANS-CONT processing option is set to BLOCK, the second column shows the delimiting character when the processing option is set to MSG, and the third shows the delimiting character when TRANS is in effect. (There are no delimiting characters for CONT, because solicitation continues indefinitely.)

The last three columns show the communication-control characters added to the beginning and end of the user-supplied data when a WRITE macro instruction is issued. The first of these three columns shows the beginning and ending characters that are inserted when the RPL's option code is set to BLK, or if a WRITE LDO is being used by DO. The next shows the characters inserted when the LBM option code is in effect or a WRITELBM LDO is used. The last column applies to the LBT option code or WRITELBT LDO.

Start–Stop Devices	Soliciting: Specified as PROC = BLOCK	MSG	TRANS	Writing ((b) = inserted at the beginning, (e) = inserted at end): Specified as OPTCD = BLK	LBM	LBT
IBM 1050 Data Communication System	EOB	EOT	EOT	EOA(b) EOB(e)	EOA(b) EOB(e) [1]	EOA(b) EOB(e) [1]
IBM 2740 Communication Terminal, Model 1	EOT	EOT	EOT	EOA(b) NUL(e)	EOA(b) NUL(e)	EOA(b) NUL(e)
IBM 2740 Communication Terminal, Model 1, with checking	EOB	EOT	EOT	EOA(b) EOB(e)	EOA(b) EOB(e) [1]	EOA(b) EOB(e) [1]
IBM 2740 Communication Terminal, Model 1, with checking and station control	EOB	EOT	EOT	EOA(b) EOB(e)	EOA(b) EOT(e)	EOA(b) EOT(e)
IBM 2740 Communication Terminal, Model 2	EOT	EOT	EOT	EOA(b) EOT(e) [1]	EOA(b) EOT(e) [1]	EOA(b) EOT(e) [1]
IBM 2741 Communication Terminal	EOT	EOT	EOT	EOA(b) NUL(e)	EOA(b) NUL(e)	EOA(b) NUL(e)
IBM Communication Magnetic Card Selectric Typewriter	EOT	EOT	EOT	EOA(b) NUL(e)	EOA(b) NUL(e)	EOA(b) NUL(e)
IBM World Trade Telegraph Station	EOT	EOT	EOT	CCITT header	CCITT header	CCITT header
IBM SYSTEM/7	EOT	EOT	EOT	EOA(b) NUL(e)	EOA(b) NUL(e)	EOA(b) NUL(e)
AT&T 83B3 Selective Calling Station	EOT	EOT	EOT	none(b) EOM(e)	none(b) EOM(e)	none(b) EOM(e)
AT&T Teletypewriter Terminal, Models 33 and 35	EOT	EOT	EOT	none(b) EOM(e)	none(b) EOM(e)	none(b) EOM(e)
Western Union Plan 115A Station	EOT	EOT	EOT	none(b) EOM(e)	none(b) EOM(e)	none(b) EOM(e)

[1] And an EOT is sent when the block is acknowledged by the system with a positive response.

Figure B-1. Communication-Control Characters Used with Start-Stop Devices

Binary Synchronous Communication Devices	Soliciting: Specified as PROC= BLOCK	MSG	TRANS	Writing: (b) = inserted at the beginning, (e) = inserted at end; Specified as OPTCD= BLK	LBM	LBT
IBM 2770 Data Communication System	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) 1
IBM 2780 Data Transmission Terminal	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) 1
IBM 2972 General Banking Terminal, Models 8 and 11	ETB	ETX	EOT	STX(b) ETX(e)	STX(b) ETX(e)	STX(b) ETX(e) 1
IBM 3270 Information Display System, remotely attached	2	EOT	EOT	2, 3	3	STX(b) ETX(e) 3
IBM 3735 Programmable Buffered Terminal	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) 1
IBM 3740 Data Entry System	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) 1
IBM 3780 Data Transmission Terminal	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) 1
IBM SYSTEM/3 or IBM SYSTEM/32	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) 1
IBM SYSTEM/370	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) 1

1 And an EOT is sent when the block is acknowledged by the system with a positive response.

2 PROC=BLOCK and OPTCD=BLK are invalid for 3270 devices.

3 No line control characters are sent.

Figure B-2. Communication-Control Characters Used with BSC Devices

APPENDIX C. RETURN CODES FOR RPL-BASED MACRO INSTRUCTIONS

Return Code Posting

VTAM posts return code information in registers 0 and 15 and in certain fields of the request's RPL. These fields are referred to as the feedback fields. The manner in which registers 0, 15, and the feedback fields are posted depends on whether synchronous request handling, asynchronous request handling (with an ECB), or asynchronous request handling (with an RPL exit routine) is used. Chapter 5 of *VTAM Concepts and Planning* and Chapter 3 of *VTAM Macro Language Guide* illustrate these three methods of request handling. The following three figures parallel those in the other books (although the posting of register 15 and feedback fields has been emphasized here).

In Figure C-1, the application program issues a SEND macro instruction and specifies *synchronous* request handling. Control passes from the application program and is not returned until the operation is completed. At that time, registers 0 and 15 are set by VTAM (or, if the LERAD or SYNAD exit routine is scheduled, registers 0 and 15 are set by the exit routine) to indicate how the operation is completed. Feedback fields in the RPL are also set.

In Figure C-2, the application program issues another SEND, this time specifying asynchronous request handling and an ECB. VTAM receives control, screens the request, schedules the requested operation (if the request is in order), and returns control to the application program. This is the 'initial completion' of the asynchronous request—that is, the point at which the request is accepted or rejected. If the request was unacceptable, VTAM (or the LERAD or SYNAD exit routine, if one was available to be scheduled) indicates this in registers 0 and 15. No additional return codes are posted in RPL2's feedback fields, and no CHECK macro instruction should be issued for RPL2. (The RPL is not set active if the request is not accepted, and CHECK cannot be used to check an inactive RPL.)

The application program will find a return code of 0 in register 15 if the request was accepted. Since an ECB was specified for the request, the application program must eventually issue a CHECK macro instruction for RPL2. (A system WAIT macro instruction could be used instead, although CHECK would still be required eventually to set the RPL inactive.) When the SEND operation is eventually completed, the ECB is posted and control is again returned to the application program. This time registers 0 and 15 and RPL2's feedback fields are set by VTAM (or by the LERAD or SYNAD exit-routine, if one was invoked as the result of issuing the CHECK macro instruction) to indicate how the SEND operation was completed.

In Figure C-3, the application program again issues an asynchronous SEND request, but this time with an RPL exit routine specified instead of an ECB. As before, a zero or non zero return code is returned to the application program at initial completion, indicating that the request has been accepted or rejected. The final completion of the SEND operation results in the invocation of the RPL exit routine if the request was accepted. After CHECK has been issued, EXITRTN finds that registers 0 and 15 and the feedback fields of RPL3 have been posted.

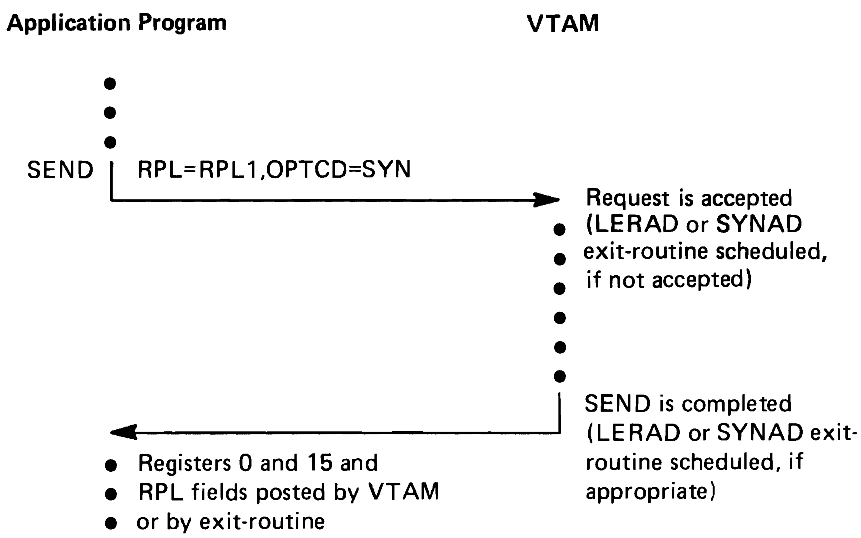


Figure C-1. Posting Return Codes for Synchronous Requests

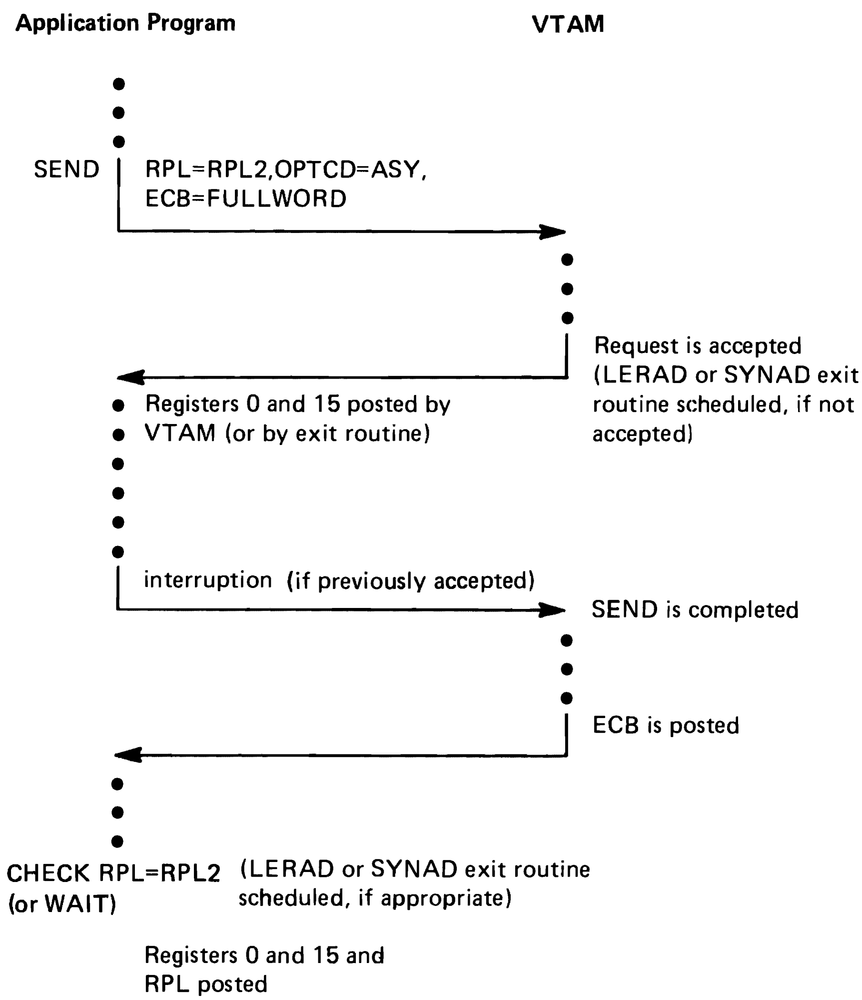


Figure C-2. Posting Return Codes for Asynchronous Requests (with CHECK)

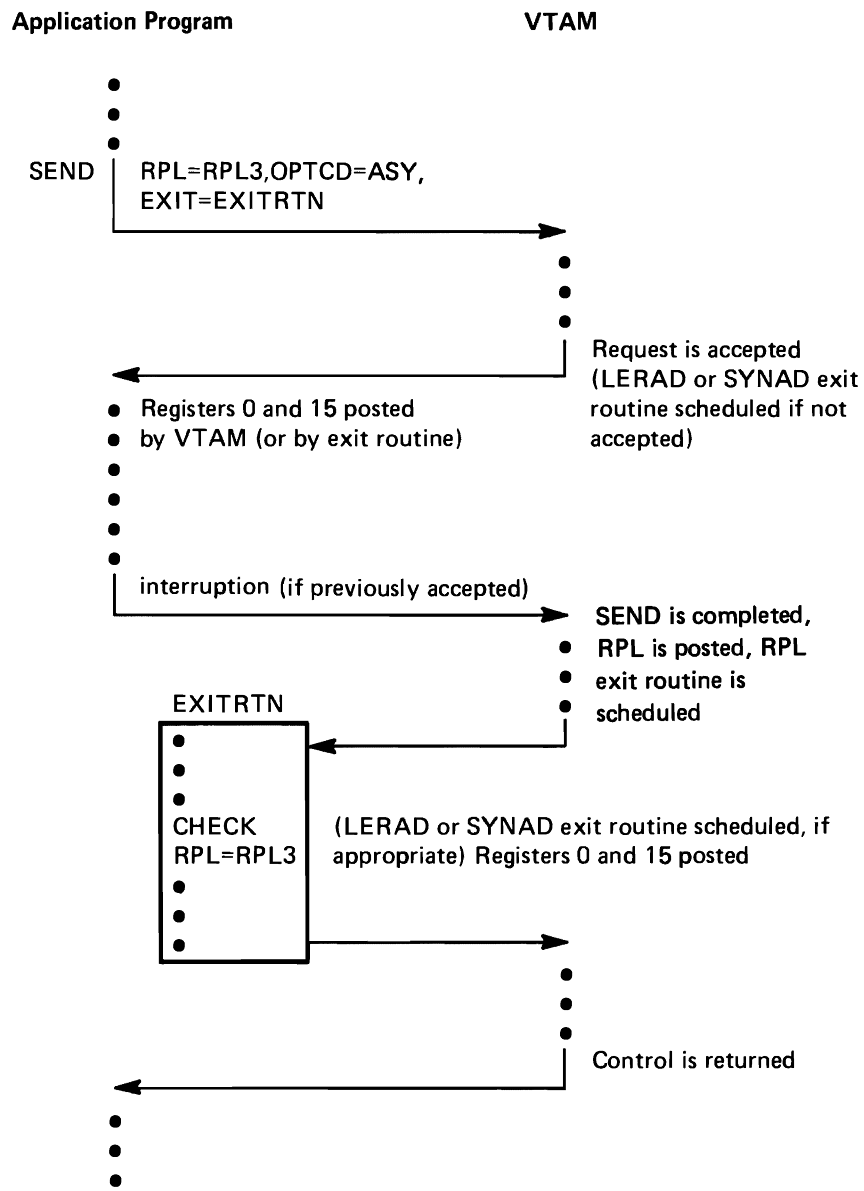


Figure C-3. Posting Return Codes for Asynchronous Requests (with an RPL Exit Routine)

Figure C-4. Completion Conditions Applicable for Initial Completion of Asynchronous Requests

Completion Condition	Explanation	Example	Registers at NSI when SYNAD-LERAD not available	Registers at entry when SYNAD-LERAD are available	Registers at NSI when SYNAD-LERAD are available	RPL feedback fields set	Applicable macros	Recommended Programmer Action (in LERAD, SYNAD, or after next sequential instruction)
Request accepted (General return code = 0, Recovery action return code = 0)	VTAM has accepted the asynchronous request and will process it. Completion information will again be available when the request is completed.	———	*Reg 15* 0 *Reg 0* 0	*Reg 15* 0 *Reg 0* 0 (LERAD-SYNAD not entered)	*Reg 15* 0 *Reg 0* 0 (LERAD-SYNAD not entered)	RTNCD=0, FDBK2=0	All RPL-based macros	———
Request not accepted (General return code = 4), Retry appropriate (Recovery action return code = 8)	VTAM has rejected the asynchronous request because of a temporary condition.	A temporary storage shortage has occurred.	*Reg 15* 4 *Reg 0* 8	*Reg 15* Address of SYNAD exit-routine *Reg 0* 8	*Reg 15* Set by SYNAD exit-routine *Reg 0*	RTNCD=8, FDBK2=0	All RPL-based macros	Issue an EXECRPL macro to retry the request.
Environment Error (Recovery action return code = 16)	VTAM has rejected the asynchronous request because of an environmental condition beyond the control of the application program.	VTAM not active.	*Reg 15* 4 *Reg 0* 16	*Reg 15* Address of SYNAD exit-routine *Reg 0* 16	*Reg 15* Set by SYNAD exit-routine *Reg 0*	RTNCD=16, FDBK2= specific error return code (13 or 14)	All RPL-based macros	Request external intervention (from the network operator, for example) or suspend processing.
Logical Error (Recovery action return code = 20)	VTAM has rejected the asynchronous request because the request violates the requirements defined in this manual.	A READ has been issued for an output-only device.	*Reg 15* 4 *Reg 0* 20	*Reg 15* Address of LERAD exit-routine *Reg 0* 20	*Reg 15* Set by LERAD exit-routine *Reg 0*	RTNCD=20, FDBK2= specific error return code (0, 2, or 3)	All RPL-based macros	Obtain a program dump and correct the program.
Logical Error with Invalid RPL (Recovery action return code = 24)	VTAM has rejected the asynchronous request because the RPL address points to an active RPL or does not point to any RPL.	An attempt was made to reuse an RPL to which no CHECK had been issued.	*Reg 15* 4 *Reg 0* 24	*Reg 15* Address of LERAD exit-routine *Reg 0* 24	*Reg 15* Set by LERAD exit-routine *Reg 0*	RPL not set and should not be examined.	All RPL-based macros	Obtain a program dump and correct the program.
Request not accepted because of a prior failure OPEN (General return code = 32, no Recovery action return code)	The RPL points to an ACB that has not been properly opened or that has been closed.	———	*Reg 15* 32 *Reg 0* Request code (see description of RPL's REQ field)	*Reg 15* 32 *Reg 0* Request code (see description of RPL's REQ field) (LERAD-SYNAD not entered)	*Reg 15* 32 *Reg 0* Request code (see description of RPL's REQ field) (LERAD-SYNAD not entered)	RPL not set.	All RPL-based macros	Obtain a program dump and correct the program.

Figure C-5 (Part 1 of 2). Completion Conditions Applicable for Completion of Synchronous Requests or for CHECK (after Asynchronous Requests)

Completion Condition	Explanation	Example	Registers at NSI when SYNAD-LERAD not available	Registers at entry when SYNAD-LERAD are available	Registers at NSI when SYNAD-LERAD are available	RPL feedback fields set	Applicable macros	Recommended Programmer Action (in LERAD, SYNAD, or after next sequential instruction
Normal Completion (General return code = 0)	The request has been completed successfully. For some macros (see right-most column), Register 0 contains additional information.	INQUIRE was issued to obtain a logon message, and there was none.	*Reg 15* 0 *Reg 0* Available information return code	*Reg 15* 0 *Reg 0* Additional information return code (LERAD-SYNAD not entered)	*Reg 15* 0 *Reg 0* Additional information return code (LERAD-SYNAD not entered)	RTNCD=0 FDBK2= Additional information return code	All RPL-based macros	RESET, WRITE, INQUIRE, INTRPRET, OPNDST, RECEIVE, and SIMLOGON macros can be completed normally with special conditions present. If these conditions are meaningful to the application program, check Register 0 or FDBK2. These conditions are explained after Fig. C-6.
Abnormal Completion (General return code = 4),								
Special Condition (Recovery action return code = 4)	The terminal has returned a special condition indicator for a request that would otherwise have been completed normally.	An exception message has been received.	*Reg 15* 4 *Reg 0* 4	*Reg 15* Address of SYNAD exit-routine *Reg 0* 4	*Reg 15* Set by SYNAD exit-routine *Reg 0*	RTNCD=4, FDBK2= special condition indicator	OPNDST, DO, READ, WRITE, SEND, and RECEIVE	Take whatever action is appropriate for the FDBK2 indicator. These indicators are explained after Fig. C-6.
Retry Appropriate (Recovery action return code = 8)	VTAM cannot complete the request because of a temporary condition.	A temporary storage shortage has occurred.	*Reg 15* 4 *Reg 0* 8	*Reg 15* Address of SYNAD exit-routine *Reg 0* 8	*Reg 15* Set by SYNAD exit-routine *Reg 0*	RTNCD=8, FDBK2= specific error return code.	All RPL-based macros	Issue an EXECRPL macro to retry the request.
Data Integrity Damaged (Recovery action return code = 12)	VTAM cannot complete the request. Either the data itself has been lost and must be resent, or the output medium (the form in a printer, for example) has been overwritten and the operation must be redone.	A hardware error occurred during an output operation.	*Reg 15* 4 *Reg 0* 12	*Reg 15* Address of SYNAD exit-routine *Reg 0* 12	*Reg 15* Set by SYNAD exit-routine *Reg 0*	RTNCD=12, FDBK2= specific error return code	OPNDST, CHANGE, and all I/O macros	Take whatever action is appropriate to the FDBK2 return code. These codes are explained after Fig. C-6. In general, the process that was interrupted should be restarted

Completion Condition	Explanation	Example	Registers at NSI when SYNAD-LERAD not available	Registers at entry when SYNAD-LERAD are available	Registers at NSI when SYNAD-LERAD are avaialble	RPL feedback fields set	Applicable macros	Recommended Programmer Action (in LERAD, SYNAD, or after next sequential instruction
Abnormal Completion (General return code = 4), (Cont.)								
Environment Error (Recovery action return code = 16)	VTAM cannot complete the request. The problem cannot be resolved without external intervention.	VTAM not active.	*Reg 15* 4; *Reg 0* 16	*Reg 15* Address of SYNAD exit-routine; *Reg 0* 16	*Reg 15* *Reg 0* Set by SYNAD exit-routine	RTNCD=16, FDBK2= specific error return code	All RPL-based macros	Take whatever action is appropriate for the FDBK2 return code. These codes are explained after Fig. C-6. In general, the logical unit or terminal is unavailable and should either be disconnected or network operation intervention should be requested.
Logical Error (Recovery action return code = 20)	VTAM cannot complete the request because it violates the requirements defined in this manual.	An I/O request is issued for a disconnected terminal.	*Reg 15* 4; *Reg 0* 20	*Reg 15* Address of LERAD exit-routine; *Reg 0* 20	*Reg 15* *Reg 0* Set by LERAD exit-routine	RTNCD=20, FDBK2= specific error return code	All RPL-based macros	Obtain a program dump and correct the program.
Logical Error with invalid RPL (Recovery action return code = 24)	VTAM cannot complete the request because the RPL address points to an active RPL or does not point to any RPL.	The RPL address was destroyed before the request was executed.	*Reg 15* 4; *Reg 0* 24	*Reg 15* Address of LERAD exit-routine; *Reg 0* 24	*Reg 15* *Reg 0* Set by LERAD exit-routine	RPL not set and should not be examined.	All RPL-based macros	Obtain a program dump and correct the program.
Abnormal Completion because of a prior OPEN failure (General return code = 32, no Recovery action return code.	VTAM cannot complete the request because the RPL points to an ACB that has not been properly opened or that has been closed.		*Reg 15* 32; *Reg 0* Request code (see description of RPL's REQ field)	*Reg 15* 32; *Reg 0* Request code (LERAD-SYNAD not entered)	*Reg 15* 32; *Reg 0* Request code (LERAD-SYNAD not entered)	RPL not set.	All RPL-based macros	Obtain a program dump and correct the program.

Figure C-5 (Part 2 of 2). Completion Conditions Applicable for Completion of Synchronous Requests or for CHECK (after Asynchronous Requests)

Types of Return Codes

VTAM always sets register 15 to 0 if a request has been accepted or has been completed normally. Register 0 is also sometimes set for normal completion.

When a request is not accepted (Figure C-4) or is completed abnormally (Figure C-5), VTAM schedules the LERAD or SYNAD exit routine. (Figures C-4 and C-5 indicate which types of errors cause LERAD and which cause SYNAD to be scheduled.) If the LERAD or SYNAD exit routine is executed upon return of control to the next sequential instruction registers 0 and 15 contain whatever values were placed in them by the exit routine. If VTAM cannot find an exit to schedule then it sets registers 0 and 15 and returns control to the next sequential instruction.

VTAM uses only two nonzero return codes in register 15: 4 and 32 (decimal). A return code of 32 is used when a failure of an OPEN is ignored by the application program; neither SYNAD nor LERAD are involved. A return code of 4 is used for all other types of errors for which a SYNAD or LERAD exit routine was not available. The register 15 return code is termed a *general return code.*

The "other types of errors" are organized into 6 classes according to the program recovery action that is appropriate for each error. VTAM generates a *recovery action return code* for each class and places the code in register 0 when control is returned to the application program or passed to the LERAD or SYNAD exit routine. VTAM also posts the recovery action return code in the RTNCD field of the RPL. The recovery action return codes occur in increments of 4 to facilitate their use in branching tables.

Note: *The recovery action return code is posted when the request's ECB is posted; if you modify RTNCD before checking the request, VTAM does not reset the code.*

VTAM also generates a *specific error return code* that defines the exact type of error within the recovery action category. The specific error return code is placed by VTAM into the FDBK2 field of the RPL.

These return codes do not occur in increments of 4; multiply them by 4 if a branching table is to be used. The specific error return codes are described below.

To summarize:

- There are three general return codes: 0 (normal), 4 (abnormal, LERAD or SYNAD not accessible to VTAM), and 32 (abnormal, failure to detect prior OPEN failure).
- There are six recovery action return codes that apply for abnormal completion. These are posted in the RTNCD field of the RPL and in register 0. If LERAD or SYNAD are invoked, the exit routine can return its own register 0 and 15 values to the next sequential instruction.
- There are numerous specific error return codes that apply for abnormal completion. These are posted in the FDBK2 field.

Specific Error Return Codes (FDBK2)

The return code set in the FDBK2 field is meaningful only when it is considered together with the recovery action return code in the RTNCD field.

You can determine the setting of the RTNCD or FDBK2 fields with either the SHOWCB or TESTCB macro instructions. For example:

```
SHOWCB     AM=VTAM,RPL=RPL1,FIELDS=(RTNCD,FDBK2),
           AREA=WORKAREA,LENGTH=8
```

Since both RTNCD and FDBK2 have been specified in the FIELDS operand, both fields will be copied into WORKAREA. Note that WORKAREA is 8 bytes long. SHOWCB right-justifies each field in the *fullword* that you supply, and sets the first 3 bytes to 0. Since two fields are being used in this example, a 2-fullword work area is required.

A TESTCB macro instruction might look like this:

```
TESTCB        AM=VTAM,RPL=RPL1,RTNCD=12
```

The feedback fields must be tested serially, since TESTCB works only with one control block field at a time. Thus another TESTCB macro instruction would be required to test the contents of the FDBK2 field.

Figure C-6 shows the RTNCD-FDBK2 combinations that are valid for a given macro instruction. Only the RPL-based macro instructions are included, since feedback posting applies only to RPL-based macro instructions. CHECK and EXECRPL are not shown because *all* of the indicated RTNCD-FDBK2 combinations are possible upon return from them.

Although specific error return codes apply only when RTNCD contains a nonzero recovery action code, this figure includes some FDBK2 values for RTNCD=0. These are additional information codes that apply to certain normally-completing requests. These codes are explained in the text following Figure C-6.

The horizontal lines in Figure C-6 do not imply any logical grouping; they have been inserted simply for legibility.

Once you have used Figure C-6 to determine which RTNCD-FDBK2 combinations are possible for a particular macro instruction, refer to the return code descriptions below for an explanation of each RTNCD-FDBK2 combination.

Should you detect a return code during program execution *other* than one described in these figures, you should cease attempting to communicate with the terminal. You may wish to use SHOWCB macro instructions to extract the contents of the RPL fields, and you should obtain a program dump. Save your source listings and any program execution output for IBM program systems representatives.

RTNCD	FDBK2	OPNDST	CHANGE	SIMLOGON	CLSDST	SETLOGON	INQUIRE	INTRPRET	DO	SOLICIT	READ	WRITE	RESET	SEND	RECEIVE	RESETSR	SESSIONC	RCVCMD	SENDCMD
0 (X'00')	0 (X'00')	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	1 (X'01')								X				X						
	2 (X'02')								X				X						
	5 (X'05')						X	X											
	6 (X'06')														X				X
	7 (X'07')						X												
	8 (X'08')	X		X															
	9 (X'09')	X																	

Figure C-6 (Part 1 of 4). RTNCD-FDBK2 Combinations Possible for Each Macro Instruction

RTNCD	FDBK2	OPNDST	CHANGE	SIMLOGON	CLSDST	SETLOGON	INQUIRE	INTRPRET	DO	SOLICIT	READ	WRITE	RESET	SEND	RECEIVE	RESETSR	SESSIONC	RCVCMD	SENDCMD
4 (X'04')	0 (X'00')								X			X							
	1 (X'01')								X		X	X							
	2 (X'02')								X		X	X							
	3 (X'03')														X				
	4 (X'04')													X	X		X		
8 (X'08')	0 (X'00')	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
12 (X'0C')	0 (X'00')								X		X	X							
	1 (X'01')								X		X	X		X	X				
	2 (X'02')								X		X	X		X	X				
	3 (X'03')								X		X	X							
	4 (X'04')								X	X		X	X						
	5 (X'05')	X	X						X		X	X	X						
	6 (X'06')								X		X	X	X						
	7 (X'07')													X	X	X	X		
	8 (X'08')													X	X	X	X		
	9 (X'09')	X																	
	10 (X'0A')								X		X	X			X				
	11 (X'0B')		X		X				X		X	X	X	X	X	X	X		
	12 (X'0C')													X	X	X	X		
	13 (X'0D')													X					
	14 (X'0E')								X			X							
	15 (X'0F')								X			X							
16 (X'10')	0 (X'00')	X		X	X														
	1 (X'01')	X																	
	2 (X'02')				X														
	3 (X'03')	X		X		X			X		X	X		X	X				
	4 (X'04')								X		X	X	X						
	5 (X'05')	X	X						X		X	X	X	X	X	X	X		
	6 (X'06')	X	X						X		X	X	X						
	7 (X'07')								X		X	X	X	X	X		X		
	8 (X'08')								X		X	X							
	9 (X'09')													X	X	X	X		
	10 (X'0A')	X	X	X	X		X	X			X	X		X	X				
	11 (X'0B')								X		X	X							
	12 (X'0C')								X		X	X							
16 (X'10')	13 (X'0D')			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	14 (X'0E')			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	15 (X'0F')													X	X	X	X		
	0 (X'00')	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	2 (X'02')	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	3 (X'03')	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	4 (X'04')	This code applies only to check.																	
	16 (X'10')	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure C-6 (Part 2 of 4). RTNCD=FDBK2 Combinations Possible for Each Macro Instruction

RTNCD	FDBK2	OPNDST	CHANGE	SIMLOGON	CLSDST	SETLOGON	INQUIRE	INTRPRET	DO	SOLICIT	READ	WRITE	RESET	SEND	RECEIVE	RESETSR	SESSIONC	RCVCMD	SENDCMD
20 (X'14')	17 (X'11')														X				
	18 (X'12')	X	X	X	X				X	X	X	X	X	X	X	X	X		
	19 (X'13')		X	X	X		X	X	X	X	X	X	X	X	X	X	X		
	20 (X'14')								X										
	21 (X'15')								X										
	22 (X'16')								X	X									
	23 (X'17')								X		X								
	24 (X'18')								X			X							
	25 (X'19')								X			X							
	26 (X'1A')								X			X							
	27 (X'1B')											X							
	28 (X'1C')											X							
	29 (X'1D')								X										
	30 (X'1E')						X	X	X		X	X		X	X			X	X
	31 (X'1F')								X										
	33 (X'21')								X										
	35 (X'23')	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	36 (X'24')								X										
	37 (X'25')								X			X							
	39 (X'27')												X						
	40 (X'28')								X			X							
	41 (X'29')								X										
	42 (X'2A')								X										
	43 (X'2B')								X			X							
	44 (X'2C')								X			X	X						
	45 (X'2D')								X				X						
	46 (X'2E')								X	X	X								
	47 (X'2F')								X										
	48 (X'30')								X										
	49 (X'31')								X			X							
	50 (X'32')								X	X	X	X	X						
	51 (X'33')								X			X							
	52 (X'34')								X			X	X						
	53 (X'35')								X										
	54 (X'36')								X				X						
	55 (X'37')								X										
	57 (X'39')								X	X	X	X	X						
	59 (X'3B')													X					
	60 (X'3C')													X			X		
	64 (X'40')													X			X		
	65 (X'41')													X	X				

Figure C-6 (Part 3 of 4). RTNCD-FDBK2 Combinations Possible for Each Macro Instruction

RTNCD	FDBK2	OPNDST	CHANGE	SIMLOGON	CLSDST	SETLOGON	INQUIRE	INTRPRET	DO	SOLICIT	READ	WRITE	RESET	SEND	RECEIVE	RESETSR	SESSIONC	RCVCMD	SENDCMD
20 (X'14')	66 (X'42')													X					
	68 (X'44')													X					
	71 (X'47')													X					
	72 (X'48')																X		
	73 (X'49')																X		
	74 (X'4A')							X											
	75 (X'4B')	X		X	X		X	X											
	76 (X'4C')						X	X											
	77 (X'4D')							X											
	78 (X'4E')	X																	
	79 (X'4F')	X																	
	80 (X'50')	X		X															
	81 (X'51')	X		X															
	82 (X'52')	X	X	X	X		X												
	83 (X'53')	X	X	X	X		X												
	84 (X'54')	X																	
	85 (X'55')	X		X	X			X											
	86 (X'56')	X																	
	87 (X'57')	X	X																
	91 (X'5B')			X	X														
	92 (X'5C')	X		X															
	93 (X'5D')		X		X														
	94 (X'5E')				X														
	95 (X'5F')				X														
	96 (X'60')				X														
	97 (X'61')					X													
	98 (X'62')		X						X	X	X	X	X	X	X	X	X		
20 (X'14')	108 (X'6C')																	X	
	109 (X'6D')																	X	X
	110 (X'6E')																		X
	111 (X'6F')																	X	X
	112 (X'70')																	X	X
	113 (X'71')																		X
	116 (X'74')													X	X	X			

Figure C-6 (Part 4 of 4). RTNCD-FDBK2 Combinations Possible for Each Macro Instruction

Normal or Conditional Completion: With normal completion, the RTNCD of X'00' is placed in register 15 and the FDBK2 return code of X'00' is placed in register 0. With conditional completion, the RTNCD of X'00' is placed in register 15 and the FDBK2 error return code is placed in register 0.

RTNCD	FDBK2	
0	0	Normal completion or request accepted

The operation has been completed normally or the request has been accepted.

0	1	RESET (COND) issued with I/O in progress

You issued a conditional RESET (OPTCD=COND), and since an I/O operation for the terminal had already reached the data transfer stage, the I/O operation has not been canceled.

0	2	Normal completion with data

A RESET (OPTCD=COND) macro instruction was completed normally, but solicited data from the terminal already resides in VTAM's buffers. This data should be obtained with READ macro instructions.

0	5	Input area too small

You issued INQUIRE or INTRPRET and specified an input work area that is too small. VTAM has placed the reuired length (in bytes) in the RPL's RECLEN field. No data has been placed in the work area.

Obtain a work area that is at least as long as the value set in RECLEN, place the length in the AREALEN field, and reissue INQUIRE or INTRPRET.

0	6	No input available

A RECEIVE or RCVCMD with OPTCD=NQ was issued and there was no input of the specified RTYPE available to satisfy the macro instruction, or a RCVCMD with OPTCD=NQ was issued and no input was available to satisfy the macro instruction.

0	7	INQUIRE information not available

You issued INQUIRE (OPTCD=LOGONMSG) to obtain a logon message, and there is none; you issued INQUIRE (OPTCD=TERMS) for a dial-in Port (either start-stop or BSC), but there was no terminal connected; you issued INQUIRE (OPTCD=TOPLOGON) for queued logons, and there are none, or you issued INQUIRE (OPTCD=CIDXLATE) for a terminal that has not been connected.

The problem may be due to an incorrectly set NAME field in the NIB, a failure on the part of the user to create the entry during VTAM definition, or a VARY command issued by the network operator that deactivated the entry.

RTNCD	FDBK2	
0	8	OPNDST (ACQUIRE) or SIMLOGON (NQ) denied–terminal in use

You attempted to acquire a terminal using OPNDST with OPTCD=ACQUIRE or SIMLOGON. The terminal is connected to another application program, and so the request is rejected.

0	9	OPNDST (ACCEPT) denied–no logons

You attempted to accept a terminal, and you indicated that your request should be rejected if no terminal is waiting to be accepted (OPTCD=NQ). There is no logon queued for your application program, and so the request is rejected.

You reissued an OPNDST (OPTCD=ACCEPT) after a previous failure resulting from a temporary storage shortage and did not log on again. Relogon using a SIMLOGON or VARY LOGON and then reissue the OPNDST.

Unsuccessful Acceptance or Completion: When a request is not accepted or completed abnormally, the RTNCD is placed in register 0 and a specific error return code is placed by VTAM into the FDBK2 field of the RPL.

4	0	RVI received

The terminal responded to your output operation with an RVI (reverse interrupt) response. When this response is received, an error lock is set for the terminal. RVIs apply only to binary synchronous devices.

4	1	Attention or reverse break received

The terminal either responded to your output operation with an attention interruption or reverse break, or terminated its transmitted data with an attention interruption or reverse break. This bit is set only for 2741 Communication Terminals and 1050 Data Communication System Terminals.

When the attention interruption or reverse break is detected, an error lock is set for the terminal.

4	2	SENSE field set

The RPL's SENSE field has been set because a sense/status message has been received from a BSC device. Near the end of this appendix there is a brief description of the type of information provided in the SENSE field. You must refer to the component description manual of each terminal for an explantion of this information.

4	3	Exception condition for incoming message

A message has been received for which an exception condition exists. The reason for the error is contained in the RPL's SSENSEI and SSENSMI fields. All messages in the current chain that have already been received by the application program should be discarded. Issue RECEIVE macros with OPTCD=TRUNC, AREALEN=0 until CHAIN=LAST or CONTROL=CANCEL is received. No responses should be sent for any element in the rest of the chain. If a negative response has not already been sent to an element of this chain and if this request requires a response, move the input sense fields to the output sense fields and send a negative response.

RTNCD	FDBK2	
4	4	Incoming response indicates exception condition

The terminal (or some other node in the network) has sent a response indicating that an exception condition was detected for one of the messages the application program sent. The SEQNO field indicates the sequence number of the message to which the negative response applies. The SSENSEI and SSENSMI (or the USENSEI) fields indicate the reason for the error condition.

If the message is part of the chain currently being transmitted to the terminal, the application program should terminate the chain by issuing a SEND with STYPE=REQ, CONTROL=DATA, and CHAIN=LAST or a SEND with STYPE=REQ and CONTROL=CANCEL to indicate where the terminal can stop discarding the messages it has been receiving. If chain elements beyond the one in error have already been sent, the SEND (POST=RESP) macros for these messages are completed with RTNCD=12 and FDBK2=13 ('request canceled by prior exception message'). If the sequence numbers need to be reset (back to the beginning of the chain, for example) a SESSIONC with CONTROL=STSN should be issued. Use SESSIONC with CONTROL=CLEAR.

8	0	Temporary storage shortage

VTAM is temporarily unable to secure enough storage to process the request. The request can be reissued (with EXECRPL, for example), unless it is the first SETLOGON (OPTCD=START) issued after opening the ACB or it is an OPNDST (OPTCD=ACCEPT). If it is the first SETLOGON after opening the ACB, you must wait and try again later. If it is an OPNDST, you must log on again using a SIMLOGON before you can reissue the OPNDST. Failure to do so may result in the failure of the OPNDST a second time with return codes of RTNCD=0 and FDBK2=9. Note: This return code may also be posted if the application program has not specified a large enough region size (OS/VS1 only).

12 (X'0C')	0	Error lock set

An error lock has been set for the device that is the object of the I/O request. The SENSE field may contain status/sense information (if it does not, the field will be set to 0).

Error locks are set by the operating system or by the communication controller's Network Control Program when an I/O error condition is detected that prevents successful completion of the operation.

When an error lock is set for a terminal, no further I/O can be accomplished until the error lock is reset with a RESET macro instruction. If I/O requests are issued while the error lock is still set, they will remain pending until RESET is finally issued.

Two forms of RESET (OPTCD=UNCOND and OPTCD=LOCK) cause the error lock to be reset. You may prefer to use the LOCK form, which resets the error lock without canceling any pending I/O requests you may have issued since the error lock was set, or that have been queued since the error lock was set.

RTNCD	FDBK2	
12 (X'0C')	1	Terminal not usable or powered-off

This code is set when the communications controller detects either a hardware check for the terminal or a modem check for the terminal's modem, or when the dial-line disconnection occurs for a dial-in terminal. Hardware and modem checks are discussed in *IBM 3704 and 3705 Communications Controller Principles of Operation,* GC30-3004. In general, this return code means that the terminal is no longer usable and should be disconnected.

This code is received when a basic-mode terminal (except a 3270) is polled and is found to be powered-off. It is received when a basic-mode or record-mode 3270 terminal is polled and the 3271 controller to which it is attached is powered-off. If the 3271 controller is powered-on but the 3270 terminal is powered-off, no return code is returned until the terminal is powered-on. See the 3270 device-dependent considerations in Appendix I.

12 (X'0C')	2	Request canceled by test request message

This I/O operation has been canceled because the terminal operator requested connection to the Teleprocessing Online Test Executive Program (TOLTEP). If a LOSTERM exit routine is available; it has been scheduled. The terminal must be disconnected (CLSDST with OPTCD=RELEASE). Any further attempts to communicate with the terminal will be rejected by VTAM. You should reestablish connection with the terminal in the same manner in which the connection was initially established—either by acquiring the terminal or by accepting it.

12 (X'0C')	3	Buffer now emptied

Prior to the receipt of this return code, VTAM's input buffers for the terminal were exhausted. The input request that empties the final block of data from the buffer receives this return code. You may now continue communicating normally with the terminal. Refer also to the explanation for RTNCD=12 and FDBK2=4 which follows.

12 (X'0C')	4	Buffers filled

Solicited data has exhausted VTAM's buffers. All I/O operations other than READ are invalid until the buffers have been emptied. To empty the buffers, reissue READ macro instructions until RTNCD=12 and FDBK2=3 (buffers now emptied) is returned; this indicates that the last block of data has been moved into the application program.

It is possible that you are soliciting too large a unit of data—for example, you are soliciting a transmission, but the installation expected that you would never solicit more than a block at a time. Even the smallest unit of data (block) may be too large, and the only solution is for the installation to enlarge VTAM's buffer size.

The second general cause of this error is that you are not emptying VTAM's buffers at a rate approximately equal to the rate that the data is arriving. You may be issuing new SOLICIT requests without first verifying that you have obtained the last block of the previously solicited data.

RTNCD	FDBK2	
12 (X'0C')	5	NCP abended, restart successful (basic mode only)

While your request was being processed, the communications controller's network control program (NCP) abnormally terminated. It has been successfully restarted, and you can reissue your request and continue. All affected I/O requests except the last one receive this return code (the last I/O request issued before the NCP abended receives a FDBK2=6 return code).

12 (X'0C')	6	NCP abended, restart successful (final I/O request) (basic mode only)

While this request was being processed, the communications controller's NCP abended and was successfully restarted. This is the last I/O request to be canceled because of the abend. An error lock has set. Any I/O requests issued *while* the NCP is in an abend condition are queued by VTAM until the NCP is restarted. The application program should either (1) issue RESET (OPTCD=LOCK) to allow other pending I/O requests for this terminal to proceed normally, or (2) issue RESET (OPTCD=UNCOND) to cancel all other pending I/O requests so that the application program can reissue them again in sequence.

12 (X'0C')	7	Connection recovery in progress (record mode only)

Contact with the logical unit has been lost and a request was issued after connection recovery has been initiated. If a LOSTERM exit routine is available, it has been scheduled. No further communication with this terminal is possible until the terminal has been reconnected. Reconnection occurs automatically; the LOSTERM exit routine is rescheduled when the reconnection occurs.

12 (X'0C')	8	Logical unit restarted

The terminal has experienced a failure but loss of contact has not occurred. However, the terminal has in effect been disconnected from your application program. Issue CLSDST to complete the disconnection process. No RECEIVE macro instructions should be attempted prior to the disconnection. OPNDST may be attempted following the CLSDST. The application program is also informed of this condition via the LOSTERM exit routine.

12 (X'0C')	9	Device varied offline

The terminal with which you are attempting to establish connection, or the application program to which you are attempting to pass a terminal, is known to VTAM but has been (or is in the process of being) deactivated by the network operator.

12 (X'0C')	10 (X'0A')	Request canceled by RESET or RESETSR

This I/O operation has been canceled by a RESET or RESETSR macro instruction issued by another part of your application program.

RTNCD	FDBK2	
12 (X'0C')	11 (X'0B')	Request canceled by CLSDST or a CHANGE or CLSDST failed because a CLSDST has already been issued

An I/O operation has been canceled because you disconnected the terminal after issuing the I/O request. CLSDST always cancels any pending I/O requests for the disconnected terminal, and returns this return code in the I/O request's RPL.

A CHANGE or CLSDST macro instruction failed because a previous CLSDST for the terminal already completed successfully; therefore, the terminal cannot be changed or disconnected again.

12 (X'0C') 12 (X'0C') Request canceled by Clear command

While the request was being processed, a Clear command was sent to the terminal. This stops all data flow and cancels all pending I/O requests for the terminal. The Clear command may have been sent by your application program (SESSIONC macro), or the command may have been sent on behalf of your application program by VTAM or by the network operator (VARY INACT command).

12 (X'0C') 13 (X'0D') Request canceled by a prior exception message

A series of chained messages was being sent to the terminal and a negative response was returned for one of them. All subsequent SEND macro instructions for that chain are canceled with this return code, until a SEND macro specifies CONTROL= CANCEL or CHAIN=LAST. If the SEND macro indicates CHAIN=LAST, the end of chain is indicated to the logical unit and the SEND macro is posted with 0C 0D. If CONTROL=CANCEL, a CANCEL command will be sent to the logical unit and the SEND will be posted with a normal return code.

12 (X'0C') 14 (X'0E') Yielded to contention

You attempted to write to a terminal on a point-to-point contention line at the same time that the terminal attempted to gain control of the line. The system yielded the line to the terminal, and your output operation has been canceled.

12 (X'0C') 15 ('0F') Yielded to contention (error lock set)

Following connection, you attempted to write to a terminal on a point-to-point contention line at the same time that the terminal attempted to gain control of the line. The system yielded the line to the terminal, and your output operation has been canceled. The error lock is set; issue RESET before attempting further communication.

16 (X'10') 0 Terminal or application program not available

The terminal with which you are attempting to establish connection, or the application program to which you are attempting to pass a terminal is known to VTAM but has been (or is in the process of being) deactivated by the network operator.

If you are attempting to acquire a terminal, the terminal is unavailable, and your application program will have to proceed without it. If you are using a NIB list, no connections will have been established. If you are attempting to pass a terminal, the receiving application program is unavailable. Your action in the latter case depends on why you are attempting to pass the terminal to that application program. For example, if you are doing so in response to a terminal operator's request for reconnection, you will probably want to notify the terminal operator that the application program is unavailable.

RTNCD	FDBK2	
16 (X'10')	1	OPNDST failed for logical unit

The OPNDST failed because no network path could be obtained, because a dial connection was not completed, because a negative response to a Bind was received, or because the logical unit does not exist. The SSENSEI and SSENSMI fields are set (these fields are described at the end of this appendix).

16 (X'10')	2	Application program does not accept logons

You attempted to disconnect a terminal and pass it to another application program, but logons cannot be queued for that application program. Logon queuing is not permitted because the application program issued SETLOGON with OPTCD=QUIESCE (indicating it no longer accepts logons) or it opened its ACB with MACRF=NLOGON (indicating it never accepts logons).

16 (X'10')	3	HALT (quick) issued

The network operator has issued a HALT command, initiating a quick closedown. You cannot connect the terminal to your application program.

A quick closedown means that you can no longer communicate with any terminals, and you should close the ACB. If you have a TPEND exit routine, it was invoked.

If a LOSTERM exit routine is available, it has been scheduled.

16 (X'10')	4	VTAM/NCP incompatibility

The I/O macro instruction failed because of incompatibilities between the output of the VTAM definition process and the Network Control Program generation process.

This problem can only be resolved by the installation.

The incompatibility probably exists because of a modification and regeneration of the Network Control Program that was accomplished without a corresponding redefinition of VTAM.

Before you can successfully attempt I/O with the terminal, the installation must remove the inconsistency that resulted in this return code. The most straightforward way of accomplishing this is to redefine VTAM, using the same input that was used to generate the *current* network control program.

16 (X'10')	5	Permanent channel or link failure

Either a permanent channel failure occurred in the channel that connects VTAM to the communications controller or to the control unit of a locally attached 3270 Information Display System or 3790 Data Communication System or a permanent link failure occurred on the link that connects VTAM to the remotely-attached communications controller. You can no longer communicate with this terminal.

16 (X'10')	6	Automatic NCP shutdown

The communications controller's network control program has shut down for one of two reasons; either the network operator has manually initiated shutdown from the communication controller panel, or the Network Control Program did not receive a response from the operating system within the time limit specified during network control program generation. You can no longer communicate with this terminal.

RTNCD	FDBK2	
16 (X'10')	7	Request canceled by VARY command

The I/O operation has been canceled because the network operator deactivated the terminal or a SESSIONC OPTCD=CLEAR macro instruction was issued after an unconditional terminate was received from the terminal. If a LOSTERM exit routine is available, it has been scheduled. You can no longer communicate with the terminal, and you should issue CLSDST to disconnect it from your application program.

16 (X'10')	8	Dial-line disconnection

A dial-line disconnection occurred for a BSC or start-stop terminal after the macro instruction began execution, but before the I/O operation itself was initiated. If a LOSTERM exit routine is available, it has been scheduled. If data is present in VTAM buffers, READ macro instructions can be issued to move the data into the application program.

If the terminal is a dial-out terminal, reissuing the I/O macro instruction may cause contact to be reestablished. If a LOSTERM exit routine is available, it is scheduled. If the terminal is a dial-in terminal, or if contact cannot be established for a dial-out terminal, issue CLSDST for the terminal.

16 (X'10')	9	Unconditional Terminate Self or character-coded logoff received

The logical unit has sent an unconditional Terminate Self command or character-coded logoff, which is a request for disconnection. No further communication with the terminal is possible. CLSDST must be issued.

16 (X'10')	10 (X'0A')	VTAM error

An error occurred in VTAM itself. No further attempts to connect or disconnect the terminal should be made.

16 (X'10')	11 (X'0B')	Dial-line disconnection (dial-out terminal)

A dial-line disconnection occurred for a *dial-out*, BSC or start-stop terminal while the I/O request was being processed. If reissuing this request does not cause contact to be reestablished, CLSDST should be issued for the terminal.

Note that this return code applies to dial-in disconnections that occur *while* the I/O operation is in progress. When the dial-line disconnection occurs after the macro instruction begins execution but before the I/O operation is initiated—that is, while VTAM is waiting for a previous I/O operation to be completed—a RTNCD value of 16 (decimal) and a FDBK2 value of 8 are returned.

16 (X'10')	12 (X'0C')	Dial-line disconnection (dial-in terminal)

A dial-line disconnection occurred for a *dial-in,* BSC or start-stop terminal while the I/O request was being processed. CLSDST should be issued for the terminal.

16 (X'10')	13 (X'0D')	VTAM inactive to your ACB

The link between VTAM and your application program (ACB) that was established with OPEN has been broken. This may have occurred because you have elsewhere issued a CLOSE that has not yet completed, or it may have occurred because VTAM has become inactive.

16 (X'10')	14 (X'0E')	Abend for program's TCB

An abend condition occurred for the user's task control block (TCB). The request was not accepted, no ECB has been posted, and no RPL exit-routine has been scheduled.

RTNCD	FDBK2	
16 (X'10')	15 (X'0F')	Buffers filled for record devices

An I/O request was issued for a logical unit, but VTAM's buffers were already filled and in the process of being cleared. The LOSTERM exit routine, if available, is scheduled.

Either the length or type of request expected is erroneous or the size of the buffer pool is too small. Make sure that the data or responses are correct, or advise your system programmer that the BUFFACT parameter of the APPL statement should indicate a larger buffer pool size.

16 (X'10')   16 (X'10')   Conditional Terminate command received

The logical unit has requested disconnection by sending a Terminate command. The application program may issue a CLSDST macro instruction to terminate the session.

16 (X'10')   17 (X'11')   SDT failure on OPNDST

A negative response was sent by a logical unit in reply to a Start Data Traffic (SDT) command. The OPNDST was not completed successfully.

20 (X'14')   0   VSAM request

The RPL contains a VSAM or other non-VTAM request code. No ECB has been posted and no RPL exit routine has been scheduled.

20 (X'14')   2   Zero EXIT field

The RPL indicates that the ECB-EXIT field is being used as an EXIT field, but the RPL exit routine address in it is 0. No RPL exit-routine has been scheduled.

20 (X'14')   3   Zero ECB field

The RPL indicates that the ECB-EXIT field is being used to point to an external ECB, but the address in the field is 0. No ECB has been posted.

20 (X'14')   4   Inactive RPL checked

CHECK was issued for an inactive RPL (an RPL that had been posted complete and for which CHECK has already been issued succesfully). All RPL-based macros must use an inactive RPL. All CHECK macros, however, must use an active RPL; an RPL cannot be checked twice.

20 (X'14')   16 (X'10')   Control block invalid

The RPL's ACB field does not contain the address of a valid ACB.

This may mean that the ACB field of the RPL was incorrectly set, the ACB has been destroyed. This return code applies only in OS/VS2.

20 (X'14')   17 (X'11')   RTYPE invalid

A RECEIVE has been issued with the RTYPE field set to NDFSYN, NDFASY, and NRESP.

20 (X'14')   18 (X'12')   CLSDST in progress

At the time this macro instruction was executed, a CLSDST request was pending for the terminal. The CLSDST request takes priority, and the request that caused this return code cannot be honored.

RTNCD	FDBK2	
20 (X'14')	19 (X'13')	CID invalid

Either the RPL's ARG field or the NIB's CID field does not contain a valid CID.

You may have inadvertently modified the field or failed to set it in the first place, or you may have used the CID of a terminal that is no longer connected to your application program.

Another possibility is that you violated the following rule: When placing a CID into the RPL's ARG field, always use the ARG keyword—ARG=(6), for example—and when placing a NIB address into the RPL's NIB field, always use the NIB keyword—for example, NIB=(6). Since these two fields occupy the same 4 bytes in the RPL, VTAM can distinguish between a NIB address and a CID only through your use of the ARG or NIB keyword. Thus the presence of this return code could mean that you placed a NIB address in the RPL with the ARG keyword, and VTAM has rejected your "CID" as invalid.

If this error code applies to CHANGE or CLSDST, you can reissue the macro instruction using the terminal's symbolic name rather than the CID. A new CID can be obtained by supplying the terminals's symbolic name to an INQUIRE (OPTCD=CIDXLATE) macro instruction.

20 (X'14')	20 (X'14')	CMD field invalid

DO encountered an LDO whose CMD field does not contain a valid command code.

Since invalid syntax (such as CMD=WRIET specified instead of CMD=WRITE) would be revealed during program assembly, you either failed to set the CMD field before issuing DO, or you modified the CMD field incorrectly before VTAM processed the DO macro instruction.

The RPL's AAREA field contains the address of the faulty LDO. If DO was processing a series of LDOs, no I/O for the previous LDOs has been initiated, even though those LDOs are valid.

20 (X'14')	21 (X'15')	WRTNRLG or WRTPRLG LDO not chained before READ

You either failed to set the FLAGS field of a WRTNRLG or WRTPRLG LDO when you should have, or you used these LDOs with PROC=MSG, TRANS, or CONT instead of BLOCK. These LDOs must be command-chained to a READ LDO and PROC must be set to BLOCK. See the LDO macro instruction description for more information.

20 (X'14')	22 (X'16')	SOLICIT issued for an output-only terminal

You issued SOLICIT (OPTCD=SPEC) for a terminal that the installation has defined as an output-only device.

20 (X'14')	23 (X'17')	READ issued for an output-only terminal

You issued a READ (OPTCD=SPEC) for a terminal that the installation defined as an output-only device.

20 (X'14')	23 (X'18')	WRITE issued for an input-only terminal

You issued a WRITE macro instruction for a terminal that the installation has defined as an input-only device.

RTNCD	FDBK2	
20 (X'14')	25 (X'19')	WRITE (ERASE) issued for an invalid terminal

You issued WRITE (OPTCD=ERASE) for the wrong type of terminal. Use this type of WRITE only to erase the screen of a display unit of a 3270 Information Display System, or of a 2265 Display Station attached to a 2270 Data Communications Terminal.

20 (X'14') 26 (X'1A') WRITE (EAU) issued for an invalid terminal

You issued WRITE (OPTCD=EAU) for the wrong type of terminal Use this type of WRITE only to erase the unprotected portion of the display unit screen of a 3270 Information Display System.

20 (X'14') 27 (X'1B') WRITE (CONV) issued for an output-only terminal

You issued WRITE (OPTCD=CONV) for a terminal that the installation has defined as an output-only terminal. This is legitimate for an ordinary WRITE (OPTCD=NCONV), but a conversational WRITE includes an input operation.

20 (X'14') 28 (X'1C') WRITE (ERASE and CONV) issued

You issued a WRITE with both the CONV option code and either the ERASE or EAU option codes specified. You cannot erase a screen with a conversational WRITE macro instruction.

20 (X'14') 29 (X'1D') COPYLBM or COPYLBT LDO chained

DO encountered a COPYLBM or COPYLBT LDO that had its FLAGS field set. You cannot set the FLAGS field of either of these LDOs (that is, you cannot command-chain them to other LDOs).

The LDO has not been processed. The RPL's AAREA field contains the address of the faulty LDO. If DO was processing a series of LDOs, those preceding the faulty LDO have been successfully processed.

20 (X'14') 30 (X'1E') Invalid data or length

You requested an input operation and either supplied an input work area address that is beyond the addressable range of your application program, or you invalidly indicated that the work area length is 0.

Check the work area address and work area length fields in the RPL for an incorrect setting. For a DO macro instruction, these are the ADDR and LEN fields. For a READ, RECEIVE, RCVCMD, INTRPRET, or INQUIRE macro instruction, these are the AREA and AREALEN fields. For a WRITE (OPTCD=CONV) macro instruction, these are the AAREA and AAREALN fields. For a SEND or SENDCMD macro instruction, these are the AREA and RECLEN fields.

20 (X'14') 31 (X'1F') LDO address invalid

You issued a DO macro instruction, and indicated an LDO address that lies beyond the addressable range of the application program. Check the AREA field of DO's RPL; you may have incorrectly modified the field, or never set an address in it before DO was executed.

RTNCD	FDBK2	
20 (X'14')	33 (X'21')	Over 100 LDOs

More than 100 LDOs were chained together.

20 (X'14')	35 (X'23')	Request type invalid

When an I/O macro instruction is issued, VTAM sets the REQ field in the RPL to indicate the type of macro instruction that is using the RPL. The presence of this return code indicates that you modified that code before the requested operation completed. To avoid this and other related errors, never modify an RPL while it is in use. Compare with "VSAM request" (RTNCD=20, FDBK2=0).

20 (X'14')	36 (X'24')	Invalid FLAGS field for a READ LDO

You misused the FLAGS field of a READ LDO. The FLAGS field cannot be used to command-chain a READ LDO to another LDO. See the description of the FLAGS operand in the LDO macro instruction for more information.

The RPL's AAREA field contains the address of the faulty LDO.

20 (X'14')	37 (X'25')	WRITE (ERASE and BLK) issued

You issued a WRITE (OPTCD=ERASE) macro instruction that also had the BLK option code specified. For a 3270 display screen, the use of BLK is never permitted in a WRITE macro instruction. For a 2770 display screen, the use of BLK together with ERASE is not permitted in the same WRITE macro instruction.

20 (X'14')	39 (X'27')	RESET option code invalid

Before VTAM completed processing the RESET macro instruction, it discovered that the COND-UNCOND-LOCK option code was not properly set. Since you cannot incorrectly set this option code using VTAM macro instructions (the macro instruction would not be assembled), you have probably modified the RPL's OPTCD field with assembler instructions and destroyed the bit settings that represent COND, UNCOND, or LOCK.

20 (X'14')	40 (X'28')	WRITE (BLK) issued

You used a WRITE macro instruction (or WRITE LDO) to send data to a display screen of a 3270 Information Display System, but the BLK option code is set. When writing to a 3270 display station, you can set the BLK-LBM-LBT option code to either LBM or LBT, but not to BLK.

20 (X'14')	41 (X'29')	READBUF used with invalid terminal

You used a READBUF LDO for an ineligible device. Use the READBUF LDO only for a display station of a 3270 Information Display System.

20 (X'14')	42 (X'2A')	COPYLBM or COPYLBT used with invalid terminal

A COPYLBM or COPYLBT LDO can only be used with a 3277 display station as the "from" device.

20 (X'14')	43 (X'2B')	WRITE (CONV) when data expected

A WRITE (OPTCD=CONV) was issued to a terminal from which data is already expected because of a previous READ, SOLICIT, or conversational WRITE operation.

RTNCD	FDBK2	
20 (X'14')	44 (X'2C')	Output not preceded by input

You used a RESET or WRITE macro instruction, or a WRITE LDO, for a 3735 Programmable Buffered Terminal without first using a SOLICIT or READ (OPTCD=SPEC) macro instruction, or a READ LDO.

The first I/O request directed at this type of terminal following connection must be a request that causes data to be solicited from the terminal.

20 (X'14') 45 (X'2D') RESET (COND) issued–error lock set

You issued a RESET (OPTCD=COND) macro instruction for a terminal, but an error lock is set for that terminal. This form of RESET cannot be used if the error lock is set. The UNCOND form of RESET is valid in this situation, however, and the LOCK form may be valid as well. See the RESET macro instruction description.

20 (X'14') 46 (X'2E') BLOCK-MSG-TRANS-CONT invalid

While processing a solicit request, VTAM discovered that the BLOCK-MSG-TRANS-CONT processing option was not properly set in the NIB when OPNDST was issued. You may have inadvertently modified this field, or used a processing option not valid for the device. See the NIB macro instruction for further information.

20 (X'14') 47 (X'2F') Too many leading graphic characters

You attempted to send too many leading graphic characters with a positive or negative response.

When you use the WRTPRLG or WRTNRLG LDOs, the ADDR and LEN fields of the LDO must indicate the address and number of leading graphic characters to be sent. The maximum number that can be sent is 15; the value you placed in the LEN field exceeds this limit.

The RPL's AAREA field contains the address of the invalid LDO.

20 (X'14') 48 (X'30') Invalid LEN (COPYLBM or COPYLBT LDOs)

You failed to properly set the LEN field of a COPYLBM or COPYLBT LDO.

When you use either of these LDOs, the ADDR field must indicate the address of a 3-byte data area. Even though the length of this area is fixed, the LEN field must nevertheless be set to 3. VTAM found a value in the LEN field that was not 3.

20 (X'14') 49 (X'31') Invalid data area

Either all or part of the output data area lies beyond the addressable range of your application program.

RTNCD	FDBK2	
20 (X'14')	50 (X'32')	Request invalid for specified device

The I/O request failed because the requested operation is invalid for the particular type of terminal to which it is directed. The error lock has been set.

This return code results when you violate these rules:

When you establish NIB processing options with OPNDST or CHANGE, do not use a processing option that is not applicable for the particular device that is the object of the OPNDST or CHANGE macro instruction. Figure 7 (at the end of the NIB macro instruction description) shows which processing options apply to each type of terminal.

Use the READ, WRITE, WRITELBM, WRITELBT, WRTHDR, WRTPRLG, and WRTNRLG LDOs only with a System/3 or System/370 CPU.

20 (X'14') 51 (X'33') WRITE canceled (input data arriving)

You attempted to send data to a terminal at the same time that data was being solicited from it. Normally, this is no problem because the write operation is suspended until the solicitation is completed. For a BSC device, however, the write operation is canceled unless the terminal has just sent a data block ending with an ETX character (that is, has just finished sending a message).

Thus for a BSC device, the receipt of this return code indicates that when the WRITE was issued, the terminal was sending data that was not the last block of a message, and the write operation has been canceled.

20 (X'14') 52 (X'34') First I/O request not READ or SOLICIT

For a dial-in BSC terminal, the first I/O request following connection must be a SOLICIT or READ (OPTCD=SPEC) macro instruction. You have used some other macro instruction as your first I/O request.

20 (X'14') 53 (X'35') Terminals not attached via same control unit

You attempted to use the COPYLBM or COPYLBT LDO to move data between two 3270 terminals that are not part of the same 3270 Information Display System. The terminals that are the objects of these LDOs must be connected to the same control unit.

You have incorrectly specified the identities of the two terminals. The ARG field of the DO macro instruction's RPL must contain the CID of the "to" terminal. The ADDR field of the LDO must contain the address of a 3-byte data area; the first byte of this data area must contain a 3270 copy control character, and the remainder of the data area must consist of the right-most two bytes of the "from" terminal's CID.

The error lock is set for the receiving terminal (the terminal whose CID is in the ARG field),but not for the "from" terminal.

20 (X'14') 54 (X'36') RESET (LOCK) invalid

You attempted to use the LOCK form of RESET in a situation in which you should have used the UNCOND form of RESET instead.

For an explanation of the restrictions that apply to the LOCK form of RESET, see the description of the RESET macro instruction (OPTCD=LOCK).

RTNCD	FDBK2	
20 (X'14')	55 (X'37')	Terminal not connected

You attempted to use the COPYLBM or COPYLBT LDO but the "from" terminal is not connected to your application program. More precisely, the "from" terminal is not connected via the ACB that you indicated in the RPL's ACB field.

20 (X'14') 57 (X'39') Invalid PROC option

VTAM completed an OPNDST normally but the setting you supplied in the NIB's field is invalid. This error is detected when the first I/O request is issued for the terminal.

20 (X'14') 59 (X'3B') NFME-NRRN response

You attempted to send a negative response with the RESPOND field set to NFME and NRRN. A response must be identified as FME, RRN, or both; in effect, you have identified the response as neither.

20 (X'14') 60 (X'3C') Previously scheduled output or request still pending

You issued a SEND (POST=SCHED), an expedited data flow control command, or SESSIONC macro instruction before the request generated by a previous one had been completed. Only one such request can be outstanding at one time. After the previous request has been completed, this macro instruction can be reissued.

20 (X'14') 64 (X'40') CONTROL invalid

You modified the bits in the CONTROL field, or you used a CONTROL value for a SESSIONC macro instruction that was not SDT, CLEAR, or STSN.

20 (X'14') 65 (X'41') Data traffic not allowed

You attempted to communicate with a logical unit to which no Start Data Traffic (SDT) command has been sent or for which a Clear is in progress. Until a logical unit is sent an SDT command, no traffic flow is possible; only SDT, Set and Test Sequence Numbers (STSN), Request Recovery (RQR), and Clear commands can be exchanged. Every time a Clear command is sent to a logical unit, a new SDT command may be required before traffic flow can resume (depends upon the transmitransmission services profile used). The error can occur on a SEND if VTAM or the network operator disconnects the logical unit.

20 (X'14') 66 (X'42') Invalid STYPE for STSN

STYPE=RESP has been specified for a SESSIONC STSN macro instruction. Only STYPE=REQ is valid.

20 (X'14') 67 (X'43') Invalid STYPE for SESSIONC

STYPE=RESP has been specified for a SESSIONC (other than STSN) macro instruction. Only STYPE=REQ is valid.

20 (X'14') 68 (X'44') RESPLIM exceeded

The number of outstanding SEND (POST=RESP) macro instructions for a terminal exceeds the RESPLIM value set in the terminal's NIB.

RTNCD	FDBK2	
20 (X'14')	71 (X'47')	3270 SEND option invalid

The RPL pointed to by your SEND macro was for a 3270 in record-mode and had one or more of the following invalid settings in effect: STYPE=RESP, RESPOND=NFME, CHAIN set to other than ONLY, or CONTROL set to other than DATA.

If the RPL was last used for a RECEIVE for the 3270, check the RESPOND field first; you may have failed to reset the field following the RECEIVE (RECEIVE sets the RESPOND field to NEX, NFME, NRRN in this case).

20 (X'14')	72 (X'48')	Redundant Clear command or invalid session control command

You attempted to send a Clear command to the terminal but no Start Data Traffic (SDT) command has been sent. Since traffic flow is already stopped, the Clear command is redundant.

You issued a SESSIONC (CONTROL=SDT or STSN) before you issued a Clear command.

20 (X'14')	73 (X'49')	Invalid STSN command

You attempted to send a Set and Test Sequence Number (STSN) command to the terminal and set the IBSQAC and/or OBSQAC fields to some value other than SET, TESTSET, IGNORE, or INVALID. See Figure 13 in the SESSIONC macro instruction description.

20 (X'14')	74 (X'4A')	Application program name not available

You issued an INTRPRET macro instruction; VTAM has located the appropriate entry in the interpret table, and found that the installation has specified a routine to provide the identity. That routine, however, has not been loaded.

20 (X'14')	75 (X'4B')	INTRPRET sequence or LOGMODE invalid

You issued an INTRPRET macro instruction but VTAM cannot locate an entry in the interpret table that corresponds to the sequence you provided.

You may have inadvertently modified the sequence or the address in the RPL's AREA field which points to the sequence. Or, the installation may have failed to properly define the entry in the interpret table.

Once your application program has been tested and debugged, and you know that none of the foregoing situations exists, you can assume this: the terminal operator or program that initiated the logon must have passed your application program an invalid logon sequence.

You issued an INQUIRE, OPNDST, SIMLOGON, or CLSDST (OPTCD=PASS) macro instruction. The NIB for this request specified a logon mode name that could not be found in the logon mode table for the logical unit named in that NIB.

RTNCD	FDBK2	
20 (X'14')	76 (X'4C')	No Terminal or application program name

You issued INQUIRE or INTRPRET, and failed to properly provide VTAM with the identity of the terminal or application program:

- INQUIRE (OPTCD=APPSTAT) was issued and the name was not that of an application program.
- INQUIRE (OPTCD=BSCID) was issued and the name was not that of a terminal containing a UTERM parameter in its TERMINAL entry.
- INQUIRE (OPTCD=TERMS) was issued and the name was not that of a cluster controller, component, terminal, line, group, local non-SNA 3270, physical unit, logical unit, or UTERM entry.
- INQUIRE (OPTCD=DEVCHAR) was issued and the name is that of an application program.
- INQUIRE (OPTCD=LOGONMSG) was issued and no logons were queued for the application program.
- INQUIRE (OPTCD=LOGONMSG) was issued and the name was not that of a terminal, component, local 3270, or logical unit.
- INQUIRE (OPTCD=SESSPARM) was issued with LOGMODE=0 in the NIB, and no logons were queued for the application program. VTAM may have deactivated the terminal between the time the logon exit routine was scheduled and the INQUIRE macro instruction was issued.
- INTRPRET was issued and the name was not that of a terminal, component, or logical unit.

Assuming that the installation properly defined the entry in the resource definition table for the terminal or application program, you have probably done one of the following: (1) failed to placed a valid CID in the RPL's ARG field; (2) failed to place a valid NIB address in the RPL's NIB field, or (3) if you did set the RPL's NIB field correctly, you failed to set a valid symbolic name in the NIB's NAME field.

20 (X'14')	77 (X'4D')	No interpret table

You issued an INTRPRET macro instruction, but there is no interpret table for the terminal. The installation may have failed to include an interpret table for this terminal during the VTAM definition process.

20 (X'14')	78 (X'4E')	Invalid use of a NIB list

You attempted to accept a terminal without setting the NIB's LISTEND field to YES.

When OPNDST (OPTCD=ACCEPT) is issued, the NIB pointed to by the RPL must have its LISTEND field set to YES. Since this is the LISTEND setting that is assumed unless you specify otherwise, you must have used the LISTEND=NO operand when you last modified the NIB.

20 (X'14')	79 (X'4F')	ACQUIRE-ACCEPT option code invalid

The OPNDST request failed because bits in the OPTCD field have been incorrectly set. The particular bits that have been incorrectly set are those that form the ACQUIRE-ACCEPT option code. This return code does not mean that the ACQUIRE option was erroneously used in place of ACCEPT, or vice versa; it means that neither ACCEPT nor ACQUIRE is indicated in the OPTCD field.

Since you cannot cause the field to be incorrectly set in this manner by using VTAM macro instructions, you may have inadvertently modified the OPTCD field with assembler instructions.

RTNCD	FDBK2	
20 (X'14')	80 (X'50')	CONANY-CONALL option code invalid

The OPNDST failed because the bits in the RPL's OPTCD field have been incorrectly set. The particular bits that have been incorrectly set are those that form the CONANY-CONALL option code. This return code does not mean that the CONANY option was erroneously used in place of CONALL, or vice versa; it means that neither CONALL nor CONANY is indicated in the OPTCD field.

Since you cannot cause the field to be incorrectly set in this manner by using VTAM macro instructions, you may have inadvertently modified the OPTCD field with assembler instructions.

20 (X'14')   81 (X'51')   Application program never accepts logons

You attempted to accept a terminal or generate a simulated logon request for a terminal, but logon request queuing cannot occur because you opened your ACB with MACRF=NLOGON.

20 (X'14')   82 (X'52')   NIB invalid

The request failed because there is no NIB at the location indicated in the RPL's NIB field.

20 (X'14')   83 (X'53')   Terminal or application program not found

The symbolic name you supplied in the NIB's NAME field or indicated via the RPL's AAREA field does not have a corresponding entry in the resource definition table. Either you failed to correctly set the NAME field, the installation did not include the entry in the resource definition table during VTAM definition, or the network operator has not activated the segment containing the name. If you were using a NIB list, no connections have been established.

20 (X'14')   84 (X'54')   Invalid terminal name

The symbolic name you supplied in the NIB's NAME field corresponds to an entry in the resource definition table, but the entry is for a node with which you cannot establish connection–such as another application program, or the Network Control Program of a communications controller. The only entries you can identify in the NAME field for OPNDST are the names of LU, TERMINAL, COMP, or LOCAL entries.

20 (X'14')   85 (X'55')   OPNDST (ACQUIRE) not authorized or application program name not available

You attempted to acquire a terminal (SIMLOGON or OPNDST) but the installation has denied you authorization to do so.

The installation may have specified during VTAM definition that your application program is not authorized to acquire any terminals. If you are authorized to acquire terminals and you still receive this return code, this means that an installation authorization routine has been invoked, and has determined that you cannot acquire the specific terminal indicated in your request.

You issued an INTRPRET macro instruction; VTAM located the appropriate entry in the interpret table and found that the installation has specified a routine to provide the identity. That routine was loaded, but it failed to provide the name of an application program. It provided a nonzero value instead.

RTNCD	FDBK2	
20 (X'14')	86 (X'56')	Invalid MODE field

You either specified MODE=BASIC for a logical unit or MODE=RECORD for a BSC or start-stop terminal.

20 (X'14')	87 (X'57')	No MODE field

You issued an OPNDST or CHANGE macro instruction and failed to set the NIB's MODE field to BASIC or RECORD.

20 (X'14')	91 (X'5B')	Invalid logon message address

The address of the logon message that you supplied in the RPL's AREA field lies beyond the addressable range of your application program.

20 (X'14')	92 (X'5C')	Duplicate terminal names

You supplied a NIB list and attempted to acquire the group of terminals represented in that list. VTAM found that at least two of the NIBs contain the same symbolic name in their NAME fields. None of the terminals have been connected to your application program.

20 (X'14')	93 (X'5D')	CLSDST invalid (terminal not connected)

The terminal represented by the CID you supplied is not connected to your application program.

This return code applies to CHANGE or CLSDST (either OPTCD=PASS or OPTCD=RELEASE) used with a terminal's CID.

You may have placed the wrong CID into the ARG field or neglected to place a CID there at all (perhaps the field still contains a CID left over from a previous CLSDST request).

20 (X'14')	94 (X'5E)	CLSDST (PASS) not authorized

CLSDST (OPTCD=PASS) is a function whose use is authorized by the installation. You attempted to use this function, but the installation has not authorized you to pass terminals to other application programs. This CLSDST macro instruction should have been issued with RELEASE in effect, not PASS.

20 (X'14')	95 (X'5F')	CLSDST (PASS) invalid

You attempted to disconnect a terminal that is not connected to your application program. This return code applies to CLSDST (OPTCD=PASS) used with a terminal's symbolic name.

(CLSDST with a terminal's symbolic name is implemented by (1) placing the address of a NIB in the NIB field of CLSDST's RPL and (2) placing the terminal's symbolic name in the NAME field of that NIB.)

20 (X'14')	96 (X'60')	CLSDST (RELEASE) invalid

You attempted to disconnect a terminal that is not connected to your application program, or had no logon queued for your application program. This return code applies to CLSDST (OPTCD=RELEASE) used with a terminal's symbolic name or with a terminal's CID.

The explanation provided for the previous return code (RTNCD value of 20 and FDBK2 value of 95) also applies to this return code when a NIB is used.

RTNCD	FDBK2	
20 (X'14')	97 (X'61')	Invalid SETLOGON

You issued SETLOGON, but the ACB's logon queue cannot be opened.

Either you opened the ACB with its MACRF field set to NLOGON or you already issued SETLOGON (OPTCD=QUIESCE) and permanently closed the logon queue. All forms of SETLOGON are thus invalid, since you are either attempting to open a logon queue that cannot be opened, or you are attempting to close a logon queue that is already closed.

20 (X'14') 98 (X'62') Wrong mode

Either you attempted to issue a basic-mode macro for a terminal that was connected with MODE=RECORD, or you attempted to use a record-mode macro for a terminal that was connected with MODE=BASIC. See the MODE operand of the NIB macro instruction.

20 (X'14') 108 (X'6C') Exceeded limit on outstanding RCVCMD requests

You attempted to issue a RCVCMD macro instruction while a previous RCVCMD was outstanding. The current limit on outstanding RCVCMD requests is one.

20 (X'14') 109 (X'6D') Application program not authorized

Your application program is not authorized by your installation to issue the SENDCMD and RCVCMD macro instructions.

20 (X'14') 110 (X'6E') Syntax error in reply to network operator message

In reply to a VTAM network operator message, you issued a SENDCMD macro instruction that contained a syntax error in the REPLY command.

20 (X'14') 111 (X'6F') SENDCMD/RCVCMD processor inactive

The portion of VTAM that processes SENDCMD and RCVCMD macro instructions is currently inactive for your application program, and the application program issued a SENDCMD or RCVCMD macro instruction. The request cannot be processed because an ACB has not been opened for the portion of the application program that issued the SENDCMD or RCVCMD or because a final CLOSE has been issued for this ACB but has not yet completed.

20 (X'14') 112 (X'70') SENDCMD/RCVCMD processor closing its ACB

The portion of VTAM that processes SENDCMD and RCVCMD macro instructions is in the process of closing its own ACB, and you (1) issued a SENDCMD macro instruction for a command other than REPLY or (2) issued a RCVCMD with OPTCD=Q and there were no VTAM messages available to satisfy the request.

20 (X'14') 113 (X'71') Operator command not valid

You attempted to send a VTAM network operator command to VTAM using the SENDCMD macro instruction; however, the command was not recognizable by VTAM or it was a command (START or HALT) that cannot be sent by the application program.

20 (X'14') 116 (X'74') Running under an SRB with no EXIT specified (OS/VS2 MVS only)

An application program running under an SRB issued a SEND, RECEIVE, or RESETSR macro instruction, but did not specify an EXIT routine.

The FDBK Field

The FDBK field is set when a READ, WRITE, DO, or INQUIRE (OPTCD=APPSTAT) macro instruction is completed successfully.

For READ, WRITE, and DO, any combination of the bits shown below may be set in the FDBK field. Although you can test the FDBK field by coding FDBK=*value* on a TESTCB macro instruction, a simpler method is available: You can code DATAFLG=UNSOL, EOB, EOM, EOT, LG, or SOH on a TESTCB macro instruction. For example:

```
TESTCB      RPL=RPL1,DATAFLG=SOH
```

An equal PSW condition code indicates that the corresponding bit is set on. The following table indicates the appropriate operand for each bit.

You may wish to use SHOWCB to examine the FDBK field. As is true with the RTNCD and FDBK2 fields, SHOWCB places the FDBK field in the right-most byte of your fullword area, and sets the high-order three bytes to zero. Thus for purposes of SHOWCB, the bit positions shown below (0, 1, 2 ... 7) become bit positions 24, 25, 26 ... 31.

For the INQUIRE macro instruction (OPTCD=APPSTAT), one of the values shown below is returned in the FDBK field. You can use either TESTCB or SHOWCB to determine the contents of FDBK. Note that although combinations of *bits* are set for READ, WRITE, and DO, a single numerical *value* is set in FDBK for INQUIRE.

FDBK Codes for READ, WRITE, and DO

Bit Position	TESTCB Operand	Explanation
0: x	DATAFLG=UNSOL	Unsolicited input. For READ (or DO, with CMD=READ), this indicates that the data was not solicited from the terminal by your application program. For WRITE (or DO, with CMD=WRITE LDO), this indicates that data was received instead of an acknowledgment, or that leading graphic characters were received with the acknowledgment.
1: . x	None	Reserved.
2: . . x	DATAFLG=EOB	End of block. For a READ or conversational WRITE, this indicates that the input data block ended with an EOB. This bit is not set if overlength data is present (which is possible if the KEEP option code is in effect). This bit is set for the READ that obtains the last portion of the overlength data, however. If this bit is set on, you will probably want to next determine if the EOM indicator is set.
3: . . . x	DATAFLG=EOM	End of message. For a READ or conversational WRITE, this indicates that the input data block ended with an ETX. This bit setting applies only to BSC devices. If this bit is set on, you will probably want to next determine if the EOT indicator is set.

4: .... x...	DATAFLG=EOT	End of transmission. For a READ or conversational WRITE, this indicates that the last block of a transmission was received. The RPL's RECLEN field must be checked; if it is set to 0, the EOT arrived unaccompanied by data. If RECLEN contains a non-zero value, a block of data (of the length indicated by RECLEN) accompanied by an EOT was received.
5: ..... x..	None	Reserved.
6: ...... x.	DATAFLG=LG	Leading graphic characters received. This bit is set only for READ, and indicates that leading graphic characters have been received (without data). This code is only set on for binary synchronous terminals.
7: x	DATAFLG=SOH	Heading block received. For a READ, this indicates that a heading block was received and placed in your input area. If RTNCD is set to 0 the data block associated with the heading block can be obtained with another READ macro instruction. If RTNCD is set to 12 (decimal), there is no associated data block–that is, the heading block arrived unacompanied by data. This code is only set on for binary synchronous terminals.

FDBK Codes for INQUIRE

FDBK Value	Explanation
0	Application program active. The application program is accepting logons.
4	Application program inactive. The application program has not yet opened its ACB.
8	Application program never accepts. The application program never accepts logons. That is, the application program opened its ACB with MACRF=NLOGON specified.
C	Application program temporarily not accepting. The application program normally accepts logons, but it has indicated that logons are not to be directed at it for the time being. That is, the application program opened its ACB with MACRF=LOGON specified, but has subsequently issued SETLOGON (OPTCD=STOP). Since the use of this type of SETLOGON implies a *temporary* closing of the logon queue, you can periodically reissue INQUIRE to determine when the application program reopens its logon queue. You will know that logons can again be directed at the application program when INQUIRE returns a value of 0 in FDBK.
10	Application program no longer accepting. The application program normally accepts logons, but it has permanently closed its logon queue. That is, it has issued SETLOGON (OPTCD=QUIESCE). No more logons can be directed at the application program. Presumably, it is about to close its ACB.

The SENSE Field (Basic Mode Only)

The SENSE field is set following READ, WRITE, and DO macro instructions when the device returns error status information. The first 2 bytes of the SENSE field contain the BSC status/sense information as received from a 3270 or 3740 terminal.

The sense field can be tested directly with the TESTCB macro instruction, or the field can be extracted with the SHOWCB macro instruction or examined with assembler instructions. SHOWCB requires that you provide a fullword work area in your application program for the SENSE field. The SENSE field contains two meaningful bytes of information; for SHOWCB, this information is right-justified in the fullword work area, and the unused portion is set to 0. The specific bits that are set in the SENSE field are identified and explained in the component description manual for the particular terminal or system. The second half of the SENSE field (the BTU response codes) cannot be examined with SHOWCB or TESTCB macro instructions; the RPL DSECT must be used (see Figure H-9).

The third and fourth bytes of the SENSE field contain the response and extended response bytes (also called NCP return codes or BTU response codes) that are forwarded by the NCP for the following terminals:

2770 Data Communication System
2780 Data Transmission Terminal
3270 Information Display System (basic-mode)
3740 Data Entry System
3780 Data Transmission Terminal

For a description of the BTU response codes, consult *IBM 3704 and 3705 Communication Controller Programmer's Reference Handbook,* GY30-3012.

The Logical Unit Sense Fields

When the application program or a logical unit receives a negative response or a Logical Unit Status (LUS) command, the response or command includes information regarding the reason for the exception condition. There are three types of information that describe the exception condition:

System sense information

System sense modifier information

User sense information

System sense information indicates one of five major classes of system-defined error.

System sense modifier information indicates one of many specific causes of the error indicated by the system sense information. Like RTNCD and FDBK2, the system sense and system sense modifier information together form a specific type of error condition within a general class of error conditions. These error conditions are described below and in the RPL DSECT (Figure H-9).

User sense information is used when the error condition is detected by the user-written program itself. No particular codes or values are defined by IBM to indicates types of errors. The node must generate its own user sense information that will be understood by the other node.

These three types of sense information—system, system modifier, and user—are set in RPL fields. Three fields (one for each type of sense information) are set by the application program when it sends a negative response or LUS command to the logical unit. Three other fields are set by VTAM when the application program receives an exception message, a negative response, or LUS command from the logical unit. These are the names of the six fields, as they would be used on a manipulative or RPL macro instruction:

	Received by the Application Program	**Sent from the Application Program**
System sense information	SSENSEI	SSENSEO
System sense modifier information	SSENSMI	SSENSMO
User sense information	USESNEI	USENSEO

System Sense Information: The values that are set in the system sense field are predefined by IBM. These values are as follows (the operands shown here are those used with a MODCB or TESTCB macro instruction; the corresponding hexadecimal value is also shown in parentheses):

System Sense Values	**Meaning**
SSENSEI=PATH (X'80')	A path error occurred. The message could not be delivered to the intended receiver because of a physical problem in the network path or an error in the system-supplied transmission header that accompanied the message. VTAM attempts to recover the path; however, if no recovery action is possible, disconnect the logical unit.
SSENSEI=CPM (X'40') SSENSEO=CPM (X'40')	An unrecoverable request header error occurred. The sender did not correctly enforce the current session protocols. Disconnect the logical unit.
SSENSEI=STATE (X'20') SSENSEO=STATE (X'20')	A state error occurred in the application program's or logical unit's use of sequence numbers, chaining commands, bracket indicators, or change-direction indicators. A state error can also occur when a data flow control command is issued or data is sent after a Clear command or when a session control command is issued before a Clear command. This type of error is recoverable; use Clear, STSN, and SDT commands.
SSENSEI=FI (X'10') SSENSEO=FO (X'10')	A request error occurred. The application program or logical unit cannot handle the message because the message itself is invalid. This error may or may not be recoverable.
SSENSEI=RR (X'08') SSENSEO=RR (X'08')	A request reject occurred. The message was delivered to the intended receiver; it was correctly interpreted, but it was not handled by the receiver. This is a recoverable condition.

System Sense Modifier Information: The values that are set in the system sense modifier field are also predefined by IBM. For the specified system sense information, the modifiers define the specific type of error that occurred. The values for the system sense modifier field are as follows (the operands shown here are those used with a MODCB or TESTCB macro instruction; the corresponding hexadecimal value is also shown in parentheses):

When the system sense field is set to:	**The system sense modifier field can be set to:**		**Meaning**
SSENSEI=PATH	SSENSMI=1	(X'01')	Intermediate controller failure. A machine or program check has occurred in an intermediate controller in the network. The current request has been discarded, and a response may or may not be possible.
	SSENSMI=2	(X'02')	Link failure. The data link has failed.
	SSENSMI=3	(X'03')	Logical unit inoperative. The logical unit is unable to process requests.
	SSENSMI=4	(X'04')	Unrecognizable DAF. An intermediate controller has no routing information for the destination address field (DAF) associated with the message, or there is no logical unit with the destination address specified.
	SSENSMI=5	(X'05')	No session. A session has not been established between the sender and receiver, or an intermediate controller does not have an active program that can establish the connection.
	SSENSMI=6	(X'06')	Invalid FID. The transmission header for the message contains an invalid format identification (FID). This may have resulted from an erroneous VTAM definition of a logical unit or physical unit.
	SSENSMI=7	(X'07')	Segmenting error. Segments have been received out of order (for example, first, last, middle), or segmenting is not supported. If segmenting is not supported, a negative response will be returned for the first segment only; the subsequent segments are discarded.

	SSENSMI=8 (X'08')	Physical unit not active. The physical unit with which the logical unit is associated has not been activated, and the current request has not resulted in a command to activate it.
	SSENSMI=9 (X'09')	Logical unit not active. The logical unit has not been activated, and the current request has not resulted in a command to activate it.
	SSENSMI=10 (X'0A')	Reserved.
	SSENSMI=11 (X'0B')	Incomplete transmission header. The transmission received was shorter than the required length for a transmission header.
	SSENSMI=12 (X'0C')	Invalid data count field. The length specified in the data count field of the transmission header does not match the actual length of the transmission.
	SSENSMI=13 (X'0D')	Lost contact. Contact with the receiving station has been lost; however, the link itself did not fail. If the reason for the loss of contact cannot be determined, it will be treated as a link failure.
SSENSEI=CPM SSENSEO=CPM	SSENSMI=1 (X'01') SSENSMO=1 (X'01')	Reserved.
	SSENSMI=2 (X'02') SSENSMO=2 (X'02')	Reserved.
	SSENSMI=3 (X'03') SSENSMO=3 (X'03')	Begin Bracket (BB) not permitted. The Begin Bracket indicator was set in a message that was in a middle or last position in a chain.
	SSENSMI=4 (X'04') SSENSMO=4 (X'04')	End Bracket (EB) not permitted. The End Bracket indicator was set in a message that was in the middle position or the first position of a multiple message chain or it was set in violation of the protocol in use.
	SSENSMI=5 (X'05') SSENSMO=5 (X'05')	Incomplete request header. The transmission received was shorter than that required for the combined transmission and request headers.

	SSENSMI=6 (X'06') SSENSMI=6 (X'06')	Exception response not permitted. An exception response was requested when not permitted.
	SSENSMI=7 (X'07') SSENSMO=7 (X'07')	Definite response not permitted. A definite response was requested when not permitted.
	SSENSMI=8 (X'08') SSENSMO=8 (X'08')	Pacing not supported. Pacing was requested but the receiver does not support pacing.
	SSENSMI=9 (X'09') SSENSMO=9 (X'09')	Change-direction not permitted. The change-direction indicator was erroneously set.
	SSENSMI=10 (X'0A') SSENSMO=10 (X'0A')	No response not permitted. No response was requested when a response was required.
	SSENSMI=11 (X'0B') SSENSMO=11 (X'0B')	Chaining invalid. Chaining has been requested when it is invalid in the current session.
	SSENSMI=12 (X'0C') SSENSMO=12 (X'0C')	Bracket protocol invalid. Bracket protocol has been requested when it is invalid in the current session.
	SSENSMI=13 (X'0D') SSENSMO=13 (X'0D')	Change-direction invalid. Change-direction has been requested when it is not supported in the current session.
	SSENSMI=14 (X'0E') SSENSMO=14 (X'0E')	Sense data included not permitted. Sense data has been requested when it is not permitted in the current session.
	SSENSMI=15 (X'0F') SSENSMO=15 (X'0F')	Format indicator not permitted. The format indicator was set when it is invalid in the current session.
	SSENSMI=16 (X'10') SSENSMO=16 (X'10')	Alternate code not permitted. The alternate code indicator was set when it is invalid in the current session.
SSENSEI=STATE SSENSEO=STATE	SSENSMI=1 (X'01') SSENSMO=1 (X'01')	Invalid sequence number. The sequence number received with a normal flow request was not one greater than the sequence number of the previous request.
	SSENSMI=2 (X'02') SSENSMO=2 (X'02')	Chaining error. Parts of the chain were received out of sequence (for example, first, middle, first).

	SSENSMI=3 SSENSMO=3	(X'03') (X'03')	Bracket error. The sender did not enforce bracket protocol correctly. This does not include contention or race errors.
	SSENSMI=4 SSENSMO=4	(X'04') (X'04')	Change-direction error. A message was received before the receiver authorized a change of direction.
	SSENSMI=5 SSENSMO=5	(X'05') (X'05')	Data traffic reset state. Data or a data flow control command received after a Clear command had been issued.
	SSENSMI=6 SSENSMO=6	(X'06') (X'06')	Data traffic quiesced. Data or a data flow control command was received from a logical unit which had already sent a Quiesce or Shutdown Complete (QC or SHUTC) or has not yet responded with a Release Quiesce (RELQ).
	SSENSMI=7 SSENSMO=7	(X'07') (X'07')	Data traffic not reset. A session control command was received and a Clear command had not been issued previously.
SSENSEI=FI SSENSEO=FI	SSENSMI=1 SSENSMO=1	(X'01') (X'01')	Message data error. User data in the message is not acceptable to the receiver (for example, it contains an unsupported character code or it contains a formatted field that is not acceptable).
	SSENSMI=2 SSENSMO=2	(X'02') (X'02')	Message length error. The message was too long or too short.
	SSENSMI=3 SSENSMO=3	(X'03') (X'03')	Function not supported. The function requested is not supported by the receiver. The function may have been requested by a formatted request code, a field in a message, or a control character.
	SSENSMI=4 SSENSMO=4	(X'04') (X'04')	Reserved.
	SSENSMI=5 SSENSMO=5	(X'05') (X'05')	Parameter error. A parameter is invalid or not supported by the receiver.
	SSENSMI=6 SSENSMO=6	(X'06') (X'06')	Reserved.

	SSENSMI=7 (X'07') SSENSMO=7 (X'07')	Category not supported. The network control commands sent are not supported by the receiver.
	SSENSMI=8 (X'08') SSENSMO=8 (X'08')	Invalid function management header. The function management header is not understood or translatable by the receiver, or an expected function management header is not present.
SSENSEI=RR SSENSEO=RR	SSENSMI=1 (X'01') SSENSMO=1 (X'01')	Resource not available. The requested logical unit, physical unit, or link is not available.
	SSENSMI=2 (X'02') SSENSMO=2 (X'02')	Operator intervention required. Operator intervention is required for example, forms or cards are required at an output device, or the device may be temporarily in local mode).
	SSENSMI=3 (X'03') SSENSMO=3 (X'03')	Missing password. The required password is missing.
	SSENSMI=4 (X'04') SSENSMO=4 (X'04')	Invalid password. The password is invalid.
	SSENSMI=5 (X'05') SSENSMO=5 (X'05')	Session limit exceeded. The session requested cannot be established because an intermediate controller or logical unit is already operating at capacity.
	SSENSMI=6 (X'06') SSENSMO=6 (X'06')	Resource unknown. The logical unit, physical unit, or link name specified in a request is not known to the receiver.
	SSENSMI=7 (X'07') SSENSMO=7 (X'07')	Reserved.
	SSENSMI=8 (X'08') SSENSMO=8 (X'08')	Reserved.
	SSENSMI=9 (X'09') SSENSMO=9 (X'09')	Mode inconsistancy. The requested receiver is not in the appropriate mode.

SSENSMI=10 (X'0A') SSENSMO=10 (X'0A')	Permission rejected. The receiver has denied an explicit or implicit request of the sender.
SSENSMI=11 (X'0B') SSENSMO=11 (X'0B')	Bracket race error. A recoverable, apparent violation of bracket protocol has occurred; this can arise when both the sender and receiver are permitted to initiate and terminate brackets.
SSENSMI=12 (X'0C') SSENSMO=12 (X'0C')	Procedure not supported. A procedure named in a message is not supported by the receiver.
SSENSMI=13 (X'0D') SSENSMO=13 (X'0D')	Reserved.
SSENSMI=14 (X'0E') SSENSMO=14 (X'0E')	Logical unit not authorized. The requesting logical unit is not authorized to use the requested resource.
SSENSMI=15 (X'0F') SSENSMO=15 (X'0F')	End user not authorized. The requesting end user is not authorized to use the requested resource.
SSENSMI=16 (X'10') SSENSMO=16 (X'10')	Missing requester identification. A required requester identification was missing in the current request.
SSENSMI=17 (X'11') SSENSMO=17 (X'11')	Break. The sender is asking the receiver to terminate the current chain with a Cancel command or a Last in Chain indicator.
SSENSMI=18 (X'12') SSENSMO=18 (X'12')	Insufficient resource. The receiver cannot act upon a request because of a temporary lack of resources.
SSENSMI=19 (X'13') SSENSMO=19 (X'13')	Bracket bid reject - no RTR. A Bid or Begin Bracket indicator been received, and the request has been rejected; a Ready to Receive (RTR) command will not be sent.
SSENSMI=20 (X'14') SSENSMO=20 (X'14')	Bracket bid reject - RTR. A Bid or Begin Bracket indicator has been received and the request has been rejected; however a Ready to Receive (RTR) command will be sent.

SSENSMI=21 (X'15') SSENSMO=21 (X'15')	Function active. A request to activate a network element or procedure has been received, but the element or procedure is already active.
SSENSMI=22 (X'16') SSENSMO=22 (X'16')	Function inactive. A request to deactivate a network element or procedure has been received, but the element or procedure was not active.
SSENSMI=23 (X'17') SSENSMO=23 (X'17')	Link inactive. A request requires the use of a link, but that link is inactive.
SSENSMI=24 (X'18') SSENSMO=24 (X'18')	Link procedure in progress. Contact, discontact, IPL, or another link procedure was in progress when the current conflicting request was received.
SSENSMI=25 (X'19') SSENSMO=25 (X'19')	RTR not required. The receiver of a Ready to Receive (RTR) command has nothing to send.
SSENSMI=26 (X'1A') SSENSMO=26 (X'1A')	Sequence of requests invalid. An invalid sequence of requests has been received.
SSENSMI=27 (X'1B') SSENSMO=27 (X'1B')	Receiver in transmit mode. The receiver cannot accept the current request because the receiver is in the half-duplex contention transmit mode.
SSENSMI=28 (X'1C') SSENSMO=28 (X'1C')	Request not executable. The requested function could not be executed because the receiver is in a permanent error condition.
SSENSMI=33 (X'21') SSENSMO=33 (X'21')	Invalid session parameters. Session parameters are not valid or not supported by a network addressable unit for which the session is requested.
SSENSMI=36 (X'24') SSENSMO=36 (X'24')	Component terminated. The device indicated by the function management header has had its session terminated because of an error condition or resource depletion.
SSENSMI=37 (X'25') SSENSMO=37 (X'25')	Component not available. The device indicated by the function management header is not available.

SSENSMI=38 (X'26') SSENSMO=38 (X'26')	Function not supported. A function requested in a function management data request unit is not supported by the receiver.
SSENSMI=39 (X'27') SSENSMO=39 (X'27')	Intermittant error, retry requested.
SSENSMI=40 (X'28') SSENSMO=40 (X'28')	Reply not allowed. A request requires a normal-flow reply, but the outbound data flow for this logical unit is quiesced or shut down, and there is no delayed reply capability.
SSENSMI=41 (X'29') SSENSMO=41 (X'29')	Reply requested (without changing direction).
SSENSMI=42 (X'2A') SSENSMO=42 (X'2A')	Presentation space altered at the secondary logical unit.
SSENSMI=43 (X'2B') SSENSMO=43 (X'2B')	Presentation space error, not caused by the operator.

User Sense Information: The values that are set in the user sense field are defined by the logic in the application program and in the customer-coded portion of the logical unit, not by IBM. The user sense field may be used to inform the application program or logical unit that an exception condition is not being raised by a SNA-defined error but because of an application program-related error or it can be used to further modify the system sense and system sense modifier values. For function management header errors (SSENSEI or SSENSEO=FI and SSENSMI or SSENSMO='08'), the user sense field may contain SNA-defined sense values.

The user sense field can also contain status and sense information if the application program is communicating with a non-SNA (basic mode) 3270. For more information see Appendix I.

When the system sense field is set to:	And the system sense modifier is set to:	The user sense field may be set to:	Meaning
SSENSEI=FI SSENSEO=FI	SSENSMI=8 (X'08') SSENSMO=8 (X'08')	USENSEI=X'4001' USENSEO=X'4001'	Invalid function management header type.
		USENSEI=X'4002' USENSEO=X'4002'	Invalid function management header function.
		USENSEI=X'4003' USENSEO=X'4002'	Compression not supported.
		USENSEI=X'4004' USENSEO=X'4002'	Compaction not supported.
		USENSEI=X'4005' USENSEO=X'4002'	Basic exchange not supported.
		USENSEI=X'4006' USENSEO=X'4002'	Only basic exchange is supported.
		USENSEI=X'4007' USENSEO=X'4007'	Media not supported.
		USENSEI=X'4008' USENSEO=X'4008'	Code selection compression violation.
		USENSEI=X'4009' USENSEO=X'4009'	Function management header C not supported.
		USENSEI=X'400A' USENSEO=X'400A'	Demand select not supported.
		USENSEI=X'400B' USENSEO=X'400B'	DSNAME not supported.
		USENSEI=X'400C' USENSEO=X'400C'	Invalid media subaddress field.
		USENSEI=X'2001' USENSEO=X'2001'	Invalid destination - active.
		USENSEI=X'2002' USENSEO=X'2002'	Invalid destination - inactive.
		USENSEI=X'2003' USENSEO=X'2003'	Invalid destination - suspended.
		USENSEI=X'2004' USENSEO=X'2004'	Invalid suspend resume sequence.
		USENSEI=X'2005' USENSEO=X'2005'	Interruption level violation.
		USENSEI=X'2006' USENSEO=X'2006'	Invalid resume priorities.
		USENSEI=X'2007' USENSEO=X'2007'	Destination not available.
		USENSEI=X'2008' USENSEO=X'2008'	Invalid end sequence.

When the system sense field is set to:	And the system sense modifier is set to:	The user sense field may be set to:	Meaning
		USENSEI=X'2009' USENSEO=X'2009'	Invalid function management header length.
		USENSEI=X'200A' USENSEO=X'200A'	Invalid field setting.
		USENSEI=X'200B' USENSEO=X'200B'	Invalid destination.
		USENSEI=X'200C' USENSEO=X'200C'	Invalid ERCL.
		USENSEI=X'200D' USENSEO=X'200D'	Invalid DST.
		USENSEI=X'200E' USENSEO=X'200E'	Invalid concatenation.
		USENSEI=X'2010' USENSEO=X'2010'	Bind function management header subset violation.
		USENSEI=X'0801' USENSEO=X'0801'	Invalid function code parameters.
		USENSEI=X'0803' USENSEO=X'0803'	Forms function cannot be performed.
		USENSEI=X'0805' USENSEO=X'0805'	Copy function cannot be performed.
		USENSEI=X'0806' USENSEO=X'0806'	Compaction table outside supported subset.
		USENSEI=X'0807' USENSEO=X'0807'	Invalid PDIR identifier.
		USENSEI=X'0808' USENSEO=X'0808'	Train function cannot be performed.
		USENSEI=X'0809' USENSEO=X'0809'	FCB load function cannot be performed.
		USENSEI=X'080A' USENSEO=X'080A'	FCB load function not supported.
		USENSEI=X'080B' USENSEO=X'080B'	Invalid compaction table name.
		USENSEI=X'080C' USENSEO=X'080C'	Invalid access.
		USENSEI=X'080D' USENSEO=X'080D'	Invalid RECLEN.
		USENSEI=X'080E' USENSEO=X'080E'	Invalid NUMRECS.
		USENSEI=X'080F' USENSEO=X'080F'	Data set in use.

When the system sense field is set to:	And the system sense modifier is set to:	The user sense field may be set to:	Meaning
		USENSEI=X'0810' USENSEO=X'0810'	Data set not found.
		USENSEI=X'0811' USENSEO=X'0811'	Invalid password.
		USENSEI=X'0812' USENSEO=X'0812'	Function not allowed for data set.
		USENSEI=X'0813' USENSEO=X'0813'	Record too long.
		USENSEI=X'0814' USENSEO=X'0814'	Data set full.
		USENSEI=X'0815' USENSEO=X'0815'	Invalid RECID.
		USENSEI=X'0817' USENSEO=X'0817'	Invalid VOLID format.
		USENSEI=X'0818' USENSEO=X'0818'	Number of logical units per chain exceeded.
		USENSEI=X'0819' USENSEO=X'0819'	Data set exists.
		USENSEI=X'081A' USENSEO=X'081A'	No space available.
		USENSEI=X'081B' USENSEO=X'081B'	Invalid VOLID.
		USENSEI=X'081C' USENSEO=X'081C'	Invalid DSACCESS.
		USENSEI=X'081D' USENSEO=X'081D'	Invalid REX type.
		USENSEI=X'081E' USENSEO=X'081E'	Insufficient resolution space.
		USENSEI=X'081F' USENSEO=X'081F'	Invalid key technique.
		USENSEI=X'0820' USENSEO=X'0820'	Invalid key displacement.
		USENSEI=X'0821' USENSEO=X'0821'	Invalid key.
		USENSEI=X'0822' USENSEO=X'0822'	Invalid N.
		USENSEI=X'0823' USENSEO=X'0823'	Invalid KEYIND.
		USENSEI=X'0824' USENSEO=X'0824'	Invalid SERID.

APPENDIX D. RETURN CODES FOR MANIPULATIVE MACRO INSTRUCTIONS

When the application program receives control from any of the manipulative macro instructions (GENCB, MODCB, TESTCB, or SHOWCB), register 15 is set to one of the values shown below.

0 The macro instruction was completed successfully.

If the macro instruction is GENCB, and the control block (or blocks) have been built in dynamically allocated storage, register 1 contains the address of the control blocks and register 0 contains their total length (in bytes).

4 An error occurred. A return code is placed in register 0 indicating the cause of the error (see Figure D-1).

8 An error occurred. Specifically, an attempt has been made to use the execute form of the macro instruction to enter a new item in the parameter list. (Only modifications to existing parameters lists are allowed, as explained in Appendix F.) Register 0 is not set.

12 A DOS/VS system control error occurred. These errors involve the READ and SIZE operands of the EXEC statement used to initiate the application program (the EXEC statement is described in *DOS/VS System Control Statements,* GC33-5376). A return code indicating the cause of the error is placed in register 0 (see Figure D-2).

When a return code of 4 or 12 is placed in register 15, an error return code is placed in register 0. Figures D-1 and D-2 explain these error return codes, and indicates which manipulative macro instructions are capable of returning each code.

	Applicable Macro Instructions				
Value	**GENCB**	**MODCB**	**SHOWCB**	**TESTCB**	**Explanation**
1	X	X	X	X	Invalid request type. When the access method processed the execute form, it found that the part of the parameter list that indicates the type of request (MODCB, SHOWCB, TESTCB, or GENCB) had been destroyed.
2	X	X	X	X	Invalid block type. You modified the list form's parameter list. When the access method processed the execute form, it found that the part of the parameter list which indicates the type of control block (ACB, EXLST, RPL, or NIB) had been destroyed.
3	X	X	X	X	Invalid keyword. You modified the list form's parameter list. When the access method processed the execute form, it found that part of the parameter list representing keyword types (FIELDS=, ERET=, etc.) had been destroyed.
4		X	X	X	Invalid block. The address specified with the ACB, EXLST, RPL, or NIB keyword did not indicate a valid ACB, EXLST, RPL, or NIB control block, respectively.

Figure D-1 (Part 1 of 2). Register 0 Return Codes for Manipulative Macros When Register 15 Is Set to 4

Value	GENCB	MODCB	SHOWCB	TESTCB	Explanation
	Applicable Macro Instructions				
5			X	X	Reserved (VSAM only)
6		X		X	Reserved (VSAM only)
7		X	X		Field nonexistent. You attempted to modify or extract a field from an exit list, but the specified field does not exist. For example, you may have specified MODCB EXLST=EXLST1,LERAD=LERADPGM in order to place a valid address (LERADPGM) in EXLST1's LERAD field. The receipt of this return code means that EXLST1 has no LERAD field; you never specified one on an EXLST or GENCB macro instruction.
8	X				Insufficient main storage. There is not enough storage in which to build the control block or blocks.
9	X		X		Insufficient program storage. The work area length you indicated with the LENGTH operand was not large enough to build the control blocks (GENCB) or to hold the control block fields (SHOWCB).
10					No address supplied (GENCB,MODCB). You attempted to generate an EXLST entry without specifying an address. For example, coding TPEND= is invalid.
11		X			RPL active. You attempted to modify an RPL that was active (it must be inactive).
12		X			ACB open. You attempted to modify an ACB after it had been opened (the ACB must not be opened when you modify it).
13		X			Reserved (VSAM only)
14	X	X		X	Invalid parameter list. You modified the list form's parameter list. When the access method processed the execute form, it found that the parameter list now indicates mutually exclusive keywords (as though you had, for example, specified BLK=RPL,ECB=ECB1, EXIT=PGM on a GENCB macro instruction.
15	X		X		Invalid alignment. The work area in your application program does not begin on a fullword boundary.
16	X	X	X	X	Invalid control block (access method invalid). You coded AM=VTAM on the macro instruction and included one or more parameters that are valid only for SAM.
17				X	No internal ECB. TESTCB (IO=COMPLETE) failed because there is no internal ECB in the RPL.

Figure D-1 (Part 2 of 2). Register 0 Return Codes for Manipulative Macros When Register 15 Is Set to 4

	Applicable Macro Instructions				
Value	**GENCB**	**MODCB**	**SHOWCB**	**TESTCB**	**Explanation**
4	X	X	X	X	DOS/VS system control error. The SIZE operand was omitted from the application program's EXEC statement.
8	X	X	X	X	DOS/VS system. An attempt was made to run in real mode.
12	X	X	X	X	DOS/VS system control error. The value of the SIZE operand does not allow enough space for VTAM modules. The SIZE specified should be 4K larger than the size required by VTAM.

Figure D-2. Register 0 Return Codes for Manipulative Macros When Register 15 Is Set to 12 (DOS/VS Only)

APPENDIX E. SUMMARY OF OPERAND SPECIFICATIONS FOR THE MANIPULATIVE MACRO INSTRUCTIONS

The first figure in the appendix (Figure E-1) deals with all of the operands of the manipulative macro instructions (GENCB, MODCB, SHOWCB, and TESTCB) that do not involve a particular control block field. The remaining figures deal exclusively with the operands you use to select the control block field or fields to be set, moved, or tested. These figures indicate which manipulative macro instructions apply for each operand and the types of values that can be coded with each operand.

For example, suppose you are interested in examining an ACB's OFLAGS field. Turning to Figure E-2, you locate the OFLAGS entry and note that TESTCB (but not SHOWCB) can be used to examine this field. Checking the OFLAGS entry further, you will note that this operand is coded in a fixed form—in this case, OFLAGS=OPEN.

The "Notation Category" and "Example" columns in Figures E-2 through E-5 do not apply to the SHOWCB macro instruction, where the control block field name is always coded after the FIELDS keyword (for example, FIELDS=PASSWD).

There are many different ways that a given operand might be coded, but the number of valid combinations is small. The valid coding combinations for each operand have been grouped under the heading "Notation Category" in Figures E-1 through E-5. Three of these categories, called the *address, quantity*, and *fixed value* categories, encompass almost all of the operands. A few operands fall into categories identified as the *name, register-indirect value* and *indirect value* categories. These notation categories are to be interpreted as follows:

Address

You can code any of the following expressions after the keyword and equal sign:

- Any expression that is valid for an A-type address constant. For example:

```
         MODCB    ACB=ACB1,APPLID=NAME1,AM=VTAM
         .
         .
         .
NAME1    DC       X'07'
         DC       CL7'INQUIRY'
```

- A register number or the label of an EQU instruction for the register, enclosed in parentheses. For example:

```
         L        3,ADRNAME
         MODCB    ACB=ACB1,APPLID=(3),AM=VTAM
         .
         .
         .
ADRNAME  DC       A(NAME1)
```

  Note: *This form is prohibited if you are using the "simple" list form of the macro instruction (MF=L). List forms are explained in Appendix F.*

- An expression of the form (S,*expr*) where *expr* is any expression valid for an S-type address constant. This form of operand is especially useful for gaining access to a control block field via a DSECT. For example, the program has already used GENCB to build an ACB in dynamically allocated storage and has placed the address of the ACB in register 7. A temporary work area MYACB contains the information that is to be placed into the ACB pointed to by register 7. The DSECT ACBMAP is used to access the information in MYACB:

```
         LA       5,MYACB
         USING    ACBMAP,5
         MODCB    ACB=(7),APPLID=(S,APPL1),AM=VTAM
         .
         .
         .
MYACB    DS       CL52
ACBMAP   DSECT
         DS       CL48
APPL1    DS       CL4
         END
```

Note: *This form is prohibited if you are using the "simple" list form of the macro instruction (MF=L).*

- An expression of the form (*,*expr*) where *expr* is any expression valid for an S-type address constant. The address specified by *expr* is indirect; that is, it is the address of a fullword that contains the operand. For example, the program has determined which APPLID address is to be used, and has primed register 5 with the appropriate displacement into APPLIST:

```
         L        7,APPLIST(5)
         MODCB    ACB=ACB1,APPLID=(*,0(7)),AM=VTAM
         .
         .
         .
APPLIST  EQU      *
         DC       A(APPL1)
         DC       A(APPL2)
```

Quantity

You can code any of the following expressions after the keyword and equal sign:

- A decimal number, or an expression that you have equated to a decimal number. For example:

```
         TESTCB   ACB=ACB1,ERROR=13,AM=VTAM
```

- A register number, or the label of an EQU instruction for the register number, enclosed in parentheses. For example:

```
         L        5,TESTVAL
         TESTCB   ACB=ACB1,ERROR=(5)
         .
         .
         .
TESTVAL  DS       F    (TESTVAL SET DURING PROGRAM
                       EXECUTION)
```

Note: *This form is prohibited if you are using the "simple" list form of the macro instruction (MF=L).*

- An expression of the form (S,*expr*) where *expr* is any expression valid for an S-type address constant. This form is especially useful for gaining access to a control block field via a DSECT. For example, the program has already used GENCB to build an ACB in dynamically allocated storage, and has placed the address of the ACB in register 7. A temporary work area MYACB contains the information to which the contents of the ERROR field in the ACB pointed to by register 7 are to be compared. The DSECT ACBMAP is used to access the information in MYACB.

```
         LA        5,MYACB
         USING     ACBMAP,5
         TESTCB    ACB=(7),ERROR=(S,ERR1),AM=VTAM
         .
         .
         .
MYACB    DS        CL52
ACBMAP   DSECT
         DS        CL30
ERR1     DS        CL1
         DS        CL21
         END
```

Note: *This form is prohibited if you are using the "simple" list form of the macro instruction (MF=L).*

- An expression of the form (*,*expr*) where *expr* is any expression valid for an S-type address constant. The address specified by *expr* is indirect; that is, it is the address of a fullword that contains the quantity for the operand. For example, the program has determined which ERROR value is to be tested and has primed register 5 with the appropriate displacement into ERRORLST:

```
         TESTCB    ACB=ACB1,ERROR=(*,ERRORLST(5))
                   ,AM=VTAM
         .
         .
         .
ERRORLST  EQU      *
BADNAME   DC       F'90'
BADPSWD   DC       F'152'
```

Fixed Value

You can code only the expressions that are specified in the macro instruction descriptions. For example:

```
         GENCB     BLK=ACB,MACRF=NLOGON,AM=VTAM
```

Name

You can code any of the following expressions:

- One to eight EBCDIC characters. For example:

```
         TESTCB    NIB=NIB1,NAME=TERM0003,AM=VTAM
```

- An expression of the form (*,*expr*) as explained above. The address specified by *expr* is indirect; that is, it is the address of a doubleword containing the name. The name must be left-justified and padded to the right with blanks if it does not occupy the entire doubleword. For example:

```
         L         7,NAMEPOOL
         ST        7,NEWNAME+4
         MODCB     NIB=NIB1,NAME=(*,NEWNAME),AM=VTAM
         .
         .
         .
NEWNAME  DC        CL8'TERM'
NAMEPOOL DC        CL4'0001'
```

Register-Indirect Value

You can code any of the following expressions:

- A register number or label of an EQU instruction for the register number, enclosed in parentheses. For example:

```
        MODCB    RPL=RPL1,ARG=(REG5)
        .
        .
        .
REG5    EQU      5
```

  Note: *This form is prohibited if you are using the "simple" list form of the macro instruction (MF=L).*

- An expression of the form (*,*expr*) as explained above. The address specified by *expr* is indirect; that is, it is the address of a fullword that contains the value. For example:

```
        MODCB    RPL=RPL1,ARG=(*,NEWCID),AM=VTAM
        .
        .
        .
NEWCID  DS       F    (NEWCID SET DURING PROGRAM
                      EXECUTION)
```

Indirect Value

You can code only the following expression:

- An expression of the form (*,*expr*) as explained above. The address specified by *expr* is indirect; that is, it is the address of a fullword that contains the value. For example:

```
        TESTCB   NIB=NIB1,DEVCHAR=(*,DEVMASK)
                 ,AM=VTAM
        .
        .
        .
DEVMASK DS       D    (DEVMASK SET DURING PROGRAM
                      EXECUTION)
```

Operand Keyword	Valid for GENCB	MODCB	SHOWCB	TESTCB	Notation Category	Example
ACB		X	X	X	Address	ACB=ACB1
AM	X	X	X	X	Fixed Value	AM=VTAM
AREA			X		Address	AREA=WORKAREA
BLK	X				Fixed Value	BLK=RPL
COPIES	X				Quantity	COPIES=7
ERET				X	Address	ERET=ERRPGM
EXLST		X	X	X	Address	EXLST=EXLST1
FIELDS			X		Fixed Value	FIELDS=(ARG,ECB)
LENGTH	X		X		Quantity	LENGTH=132
MF	X	X	X	X	See Appendix F	MF=(E,PARMLIST)
NIB		X	X	X	Address	NIB=NIB1
RPL		X	X	X	Address	RPL=RPL1
WAREA	X				Address	WAREA=WORKAREA

Figure E-1. Manipulative Macro Instruction Operands Exclusive of Control Block Field Operands

Operand Keyword	Valid for GENCB	MODCB	SHOWCB	TESTCB	Notation Category	Example
ACBLEN			X	X	Quantity	ACBLEN=(7)
AM	X				Fixed Value	AM=VTAM
APPLID	X	X	X	X	Address	APPLID=AREA
ERROR			X	X	Quantity	ERROR=13
EXLST	X	X	X	X	Address	EXLST=EXLST2
MACRF	X	X		X	Fixed Value	MACRF=LOGON
OFLAGS				X	Fixed Value	OFLAGS=OPEN
PASSWD	X	X	X	X	Address	PASSWD=PASSWD1

Figure E-2. Manipulative Macro Instruction Operands for ACB Fields

Operand Keyword	Valid for GENCB	MODCB	SHOWCB	TESTCB	Notation Category	Example
DFASY	X	X	X	X	Address	DFASY=(3)
RESP	X	X	X	X	Address	RESP=RESPEXIT
SCIP	X	X	X	X	Address	SCIP=(*,SCIPADR)
ATTN	X	X	X	X	Address	ATTN=ATTNRTN
LERAD	X	X	X	X	Address	LERAD=LERTN
LOGON	X	X	X	X	Address	LOGON=LGNRTN
LOSTERM	X	X	X	X	Address	LOSTERM=(6)
RELREQ	X	X	X	X	Address	RELREQ=(S,AREA1)
SYNAD	X	X	X	X	Address	SYNAD=(*,AREA2)
TPEND	X	X	X	X	Address	TPEND=(S,4(7))
EXLLEN			X	X	Quantity	EXLLEN=(3)

Figure E-3. Manipulative Macro Instruction Operands for EXLST Fields

Operand Keyword	Valid for GENCB	MODCB	SHOWCB	TESTCB	Notation Category	Example
AAREA	X	X	X	X	Address	AAREA=INAREA
AAREALN	X	X	X	X	Quantity	AAREALN=100
ACB	X	X	X	X	Address	ACB=ACB1
AREA	X	X	X	X	Address	AREA=(*,FLWORD)
AREALEN	X	X	X	X	Quantity	AREALEN=132
ARECLEN	X	X	X	X	Quantity	ARECLEN=(S,QUANT1)
ARG	X	X	X	X	Register-Indirect Value	ARG=(7)
BRACKET	X	X		X	Fixed Value	BRACKET=(BB,NEB)
BRANCH	X	X		X	Fixed Value	BRANCH=YES
CHAIN	X	X		X	Fixed Value	CHAIN=LAST
CHNGDIR	X	X		X	Fixed Value	CHNGDIR=(CMD,NREQ)
CODESEL	X	X		X	Fixed Value	CODESEL=STANDARD
CONTROL	X	X		X	Fixed Value	CONTROL=QEC
DATAFLG				X	Fixed Value	DATAFLG=EOM
ECB	X	X	X	X	Address	ECB=FULLWORD
EXIT	X	X	X	X	Address	EXIT=EXITRTN
FDBK			X	X	Quantity	FDBK=(4)
FDBK2			X	X	Quantity	FDBK2=128
IBSQAC	X	X		X	Fixed Value	IBSQAC=TESTPOS
IBSQVAL	X	X	X	X	Quantity	IBSQVAL=0
IO				X	Fixed Value	IO=COMPLETE
NIB	X	X	X	X	Address	NIB=NIB6
OBSQAC	X	X		X	Fixed Value	OBSQAC=INVALID
OBSQVAL	X	X	X	X	Quantity	OBSQVAL=(4)
OPTCD	X	X		X	Fixed Value	OPTCD=(SYN,SPEC)
POST	X	X		X	Fixed Value	POST=SCHED
RECLEN	X	X	X	X	Quantity	RECLEN=32
RESPOND	X	X		X	Fixed Value	RESPOND=(NEX,FME)
REQ			X	X	Quantity	REQ=VAL23
RPLLEN			X	X	Quantity	RPLLEN=(7)
RTNCD			X	X	Quantity	RTNCD=(*,0(3))
RTYPE	X	X		X	Fixed Value	RTYPE=(NDFSYN,DFASY)
SENSE			X	X	Quantity	SENSE=(REG3)
SEQNO	X	X	X	X	Quantity	SEQNO=(10)
SIGDATA	X	X	X	X	Quantity	SIGDATA=32767
SSENSEI				X	Fixed Value	SSENSEI=CPM
SSENSEO	X	X		X	Fixed Value	SSENSEO=STATE
SSENSMI			X	X	Quantity	SSENSMI=255
SSENSMO	X	X	X	X	Quantity	SSENSMO=(*,0(12))
STYPE	X	X		X	Fixed Value	STYPE=REQ
USENSEI			X	X	Quantity	USENSEI=4095
USENSEO	X	X	X	X	Quantity	USENSEO=(4)
USER			X	X	Quantity	USER=1024

Figure E-4. Manipulative Macro Instruction Operands for RPL Fields

Operand Keyword	Valid for GENCB	MODCB	SHOWCB	TESTCB	Notation Category	Example
BNDAREA	X	X	X	X	Address	BNDAREA=BNDADDR
CID			X	X	Register-Indirect Value	CID=(7)
CON				X	Fixed Value	CON=YES
DEVCHAR			X	X	Indirect Value	DEVCHAR=(*,0(5))
EXLST	X	X	X	X	Address	EXLST=(*,EXLSTADR)
LISTEND	X	X		X	Fixed Value	LISTEND=NO
LOGMODE	X	X	X	X	Fixed Value	CODESEL=STANDARD
MODE	X	X	X	X	Fixed Value	MODE=RECORD
NAME	X	X	X	X	Name	NAME=NYCTERM
NIBLEN			X	X	Quantity	NIBLEN=(8)
PROC	X	X		X	Fixed Value	PROC=(EIB,ELC)
RESPLIM	X	X	X	X	Quantity	RESPLIM=10
SDT	X	X		X	Fixed Value	SDT=SYSTEM
USERFLD	X	X	X	X	Quantity	USERFLD=(9)

Figure E-5. Manipulative Macro Instruction Operands for NIB Fields

APPENDIX F. LIST, GENERATE, AND EXECUTE FORMS OF THE MANIPULATIVE MACRO INSTRUCTIONS

The standard form of a manipulative macro instruction expands at assembly time into (1) nonexecutable code that represents the parameters you specified on the macro instruction, and (2) executable code that causes the access method to be entered when the macro instruction is executed. The nonexecutable code, called the parameter list, is assembled at the point in your application program where the macro instruction appears.

Various nonstandard forms of the manipulative macro instructions cause the assembler to:

- Build the parameter list where the macro instruction appears in your source code but assemble no executable code ("simple" list form)
- Assemble code that will build the parameter list at a location of your selection but assemble no executable code that causes the access method to be entered ("remote" list form)
- Assemble code that will build the parameter list at a location of your selection and assemble the code that causes the access method to be entered (generate form)
- Assemble code that will modify a parameter list and cause the access method to be entered during program execution (execute form)

Figure F-1 summarizes the actions of these various forms. It also indicates the types of programs that would use each form, and shows how the MF operand is used for each form.

As indicated in Figure F-1, the various nonstandard forms of the manipulative macro instructions are designated with the MF operand.

Form	During Assembly	During Execution	Useful for	Coded with
Standard	Parameter list built where macro appears in source code	Access method entered	Nonreenterable programs that are *not* sharing or modifying parameter lists	No MF operand
"Simple" List	Parameter list built where macro appears in source code	No executable code (execute form required)	Nonreenterable programs that *are* sharing or modifying parameter lists	MF=L
"Remote" List	Code assembled to build parameter list at a location you specify	Parameter list built, but access method not entered (execute form required)	*Reenterable* programs that *are* sharing or modifying parameter lists	MF=(L,address[,label])
Generate	Code assembled to (1) build parameter list at a location you specify, and (2) enter the access method	Parameter list built and access method entered	*Reenterable* programs that are *not* sharing or modifying parameter lists	MF=(G,address[,label])
Execute	Code assembled (where macro appears in the source code) to *modify* the parameter list whose address you supply	Parameter list modified and the access method entered	Programs using the list form	MF=(E,address)

Figure F-1. The Forms of the Manipulative Macro Instructions

The MF operand for the *list* form of any manipulative macro instruction is coded as follows:

MF= {L | L,address [,label])}

L
Indicates that this is the list form of the macro instruction. If you code just MF=L ("simple" list form), the parameter list is assembled in place. If you modify the parameter list during program execution, your program is not reenterable.

address
Indicates the location where you want the parameter list to be built during program execution. This area must begin on a fullword boundary and if your program is to be reenterable, must be in dynamically allocated storage. Since the assembler will build executable code that will in turn build the parameter list, the macro instruction must be in the executable portion of your program–that is, not treated as a program constant.

You can code this address in any of the forms of the "address" notation category (described in Appendix E). The notes there stating that register expressions are prohibited for the list form do *not* however apply to the MF operand; this restriction is true only for all the other operands of the macro instruction's list form. For example, MF=(L,(6)) is valid.

label
This is a unique name that is used as a label for an assembled EQU instruction. During program assembly, the assembler equates this label to the *length* (in bytes) of the parameter list that will be built during program execution. You can use this label to assure that you are obtaining enough dynamically allocated storage to hold the parameter list.

When coding *label* follow the same rules that apply to any label for an assembler instruction.

List form example:

```
LA       10,PLISTLEN      OBTAIN LENGTH OF PARAMETER LIST
GETMAIN  R,LV=(10)        OBTAIN STORAGE FOR PARAMETER LIST
LR       5,1              SAVE STORAGE ADDRESS
TESTCB   RPL=RPL1,DATAFLG=EOM,AM=VTAM,
         MF=(L,(5),PLISTLEN)
```

The MF operand for the *generate* form of any of the manipulative macro instructions is coded as follows:

MF=(G,address [,label])

G
Indicates that this is the generate form of the macro instruction.

address
Indicates the location where you want the parameter list to be built during program execution. Presumably, this will be in dynamically allocated storage. In both manner of use and manner of coding, this address is identical to the address described above for the list form.

label
Indicates the label to be used on an EQU instruction for the length of the parameter list. The function of the *label* operand and its rules for coding are identical to those described above for the list form.

Generate form example:

```
LA       10,PLISTLEN        OBTAIN LENGTH OF PARAMETER LIST
GETMAIN  R,LV=(10)          OBTAIN STORAGE FOR PARAMETER LIST
GENCB    BLK=RPL,AM=VTAM,
         MF=(G,(5),PLISTLEN)
```

The MF operand for the *execute* form of any of the manipulative macro instructions is coded as follows:

MF=(E,address)

E
Indicates that this is the execute form of the macro instruction.

address
Indicates the location of parameter list to be used by the access method.

The execute form allows you to modify the parameter list between the generation of that parameter list and the invocation of the access method routines that use the parameter list. Only the execute form provides a means for you to modify the parameter list after it has been built.

The optional operands you specify on the execute form of a particular macro instruction are converted by the assembler into code that will modify a parameter list during execution. This code can only modify—and not expand—the parameter list. If the parameter list is actually a list form (as is typically the case), never refer to a control block field in an execute form that you did not specify in the list form. If you fail to observe this rule, and thereby attempt to expand the parameter list, the execute form will not be processed successfully, and a return code of 8 will be posted in register 15.

Execute form example:

```
EFORM   MODCB        EXLST=EXLST1,LERAD=(3),AM=VTAM,
                     MF=(E,LFORM)
          .
          .
          .
LFORM   MODCB        EXLST=0,LERAD=0,MF=L,AM=VTAM
```

Optional and Required Operands

Operands that are required in the standard form of the manipulative macro instructions may be optional in the list, generate, or execute forms or may be prohibited in the execute form. The meaning of the operands, however, and the notation used to express them, are the same. The following assembler format tables (Figures F-2 through F-5) indicate which operands are required and which are optional for each form of each manipulative macro instruction. Any operand that does not appear in an assembler format table for a particular form is prohibited.

Name	*Operation*	*Operands*
[symbol]	GENCB List Form	BLK={ ACB|EXLST|RPL|NIB } , AM=VTAM [, keyword=value] ... [, COPIES={ 1|quantity }] [, WAREA=work area address , LENGTH=work area length] , MF={ L|(L, address [, label]) }
[symbol]	GENCB Generate Form	BLK={ ACB|EXLST|RPL|NIB } , AM=VTAM [, COPIES={ 1|quantity }] [, keyword=value] ... [, WAREA=work area address , LENGTH=work area length] , MF=(G, address [, label])
[symbol]	GENCB Execute Form	BLK={ ACB|EXLST|RPL|NIB } ,AM=VTAM [,keyword=value] ... [,COPIES={ 1 quantity }] [,WAREA=work area address ,LENGTH=work area length] ,MF=(E, parameter list address)

Figure F-2. Optional and Required Operands for the Nonstandard Forms of GENCB

Name	*Operation*	*Operands*
[symbol]	**MODCB** **List Form**	**AM=VTAM** **{ , ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address }** **{ , field name=new value }...** **, MF={ L|(L, address [, label]) }**
[symbol]	**MODCB** **Generate Form**	**AM=VTAM** **{ , ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address }** **{ , field name=new value }...** **, MF=(G, address [, label])**
[symbol]	**MODCB** **Execute Form**	**AM=VTAM** **{ , ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address }** **[{ , field name=new value }...]** **, MF=(E, parameter list address)**

Figure F-3. Optional and Required Operands for the Nonstandard Forms MODCB

Name	*Operation*	*Operands*
[symbol]	**SHOWCB** **List** **Form**	**AM=VTAM** **, ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address** **, FIELDS={ field name|(field name, ...) }** **, AREA=data area address** **, LENGTH=data area length** **, MF={ L|(L, address [, label]) }**
[symbol]	**SHOWCB** **Generate** **Form**	**AM=VTAM** **, ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address** **, FIELDS={ field name|(field name, ...) }** **, AREA=data area address** **, LENGTH=data area length** **, MF=(G, address [, label])**
[symbol]	**SHOWCB** **Execute** **Form**	**AM=VTAM** **, ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address** **[, AREA=data area address]** **, MF=(E, parameter list address)**

Figure F-4. Optional and Required Operands for the Nonstandard Forms of SHOWCB

Name	*Operation*	*Operands*
[symbol]	**TESTCB** **List** **Form**	**AM=VTAM** **[{ , ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address }]** **, field name=test value** **[, ERET=error routine address]** **, MF={ L|(L, address [, label]) }**
[symbol]	**TESTCB** **Generate** **Form**	**AM=VTAM** **[{ , ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address }]** **, field name=test value** **[, ERET=error routine address]** **, MF=(G, address [, label])**
[symbol]	**TESTCB** **Execute** **Form**	**AM=VTAM** **[{ , ACB=acb address** **, EXLST=exit list address** **, RPL=rpl address** **, NIB=nib address }]** **[, field name=test value]** **[, ERET=error routine address]** **, MF=(E, parameter list address)**

Figure F-5. Optional and Required Operands for the Nonstandard Forms of TESTCB

APPENDIX G. SUMMARY OF REGISTER USAGE

The following table shows what VTAM does with the general-purpose registers before it returns control to the application program at the next sequential instruction. It indicates which registers are left unchanged by the VTAM macro instructions and which ones may be modified between the time the macro instruction is executed and control is returned to the application program. The table also shows the disposition of the registers when any of the exit-routines receives control.

	Register 0	Register 1	Registers 2-12	Register 13	Register 14	Register 15
Upon return from OPEN and CLOSE macros	Unpredictable	Unpredictable	Unmodified	Unmodified [1]	Unpredictable	Return code
Upon return from RPL-based macros, including CHECK	[2]	Address of RPL	Unmodified	Unmodified [1]	Unpredictable	[2]
Upon return from GENCB	Error return code or control block length [3]	Control block address [3]	Unmodified	Unmodified [1]	Unpredictable	General return code
Upon return from SHOWCB	Error return code	Unpredictable	Unmodified	Unmodified [1]	Unpredictable	General return code
Upon return from MODCB or TESTCB	Error return code[4]	Unpredictable	Unmodified	Unmodified [1]	Unpredictable	General return code
Upon invocation of the LERAD or SYNAD exit-routine	Recovery action return code	Address of RPL	Unmodified since request issued		Return address	Address of exit-routine
Upon invocation of the other exit-routine	Unpredictable	Address of VTAM-supplied parameter list	Unpredictable		Return address	Address of exit-routine

1 Register 13 must indicate the address of an 18-word save area when the macro instruction is executed.

2 If the operation completed normally, register 15 is set to 0. (For some macros completing normally but with a special condition, register 0 is also set - - see Appendix C.) If an error occurred and the LERAD or SYNAD exit-routine has been invoked, registers 0 and 15 contain the values set in them by the exit-routine. If an error occurred and no LERAD or SYNAD exit-routine exists, VTAM sets register 15 to 4 or 32 (decimal) and places a recovery action return code in register 0.

3 When GENCB is used to build control blocks in dynamically allocated storage and GENCB is completed successfully (register 15 set to 0), register 1 contains the address of the generated control blocks and register 0 contains the length of the control blocks, in bytes. If GENCB is completed unsuccessfully (register 15 set to 4), register 0 contains an error return code and register 1 is unpredictable. If GENCB is completed unsuccessfully (register 15 set to 8), no error return code is set in register 0.

4 If SHOWCB, MODCB, or TESTCB is completed unsuccessfully (with register 15 set to 4), register 0 contains an error return code. If the macro instruction is completed unsuccessfully (with register 15 set to 8), no error return code is set in register 0. If the macro instruction is completed successfully (with register 15 set to 0), no particular value is set in register 0 (although it may have been modified by the macro instruction).

Figure G-1. Register Contents Upon Return of Control

APPENDIX H. CONTROL BLOCK FORMATS AND DSECTs

The ACB, EXLST, RPL, and NIB can be initialized, modified, and examined either with manipulative macro instructions (GENCB, MODCB, SHOWCB, TESTCB) or with assembler instructions. Manipulation via assembler instructions requires access to the internal structure of the control block, because displacements and bit settings must be incorporated into the assembler instructions. However, bit settings and displacements are subject to change from release to release; to avoid recoding assembler instructions when such changes occur, a DSECT should be used. IBM-supplied DSECTs are provided as part of the system macro library (source statement library in DOS/VS, SYS1.MACLIB in OS/VS). They are described in this appendix.

A DSECT is an overlay (map) containing labels that correspond to field displacements, bit settings, and byte values.

A *field displacement* is the displacement of a field from the beginning of the control block, as defined by the DS (or ORG) instructions in the DSECT. A *bit setting* is an assembler EQU instruction (such as LABEL1 EQU X'80') that identifies a particular bit or bits. The label could be used as the immediate data byte in a TM instruction, for example. A *byte value* is also an assembler EQU instruction (such as LABEL2 EQU X'23') that identifies a particular value in a byte. The label could be used as the immediate data byte of a CLI instruction, for example.

The general manner in which DSECTs are used (register preparation, USING instructions, etc.) is described in "The DSECT Instruction" in *OS/VS and DOS/VS Assembler Language,* GC33-4010.

A table (Figure H-1) is organized alphabetically by label name. It will help you locate a particular label in the format maps and DSECT descriptions.

The format maps in this appendix show the format of the control blocks. They provide a means by which a dump of the control block can be interpreted and they make the DSECT descriptions that accompany them more easily understood. The following formats and DSECTs are described:

Control Block		**DSECT Name and Operands***	
ACB	(for DOS/VS)	IFGACB	AM=VTAM
ACB	(for OS/VS)	IFGACB	AM=VTAM
EXLST	(for DOS/VS and OS/VS)	IFGEXLST	AM=VTAM
RPL	(for DOS/VS)	IFGRPL	AM=VTAM
RPL	(for OS/VS)	IFGRPL	AM=VTAM
RPL	(for DOS/VS and OS/VS)	ISTUSFBC	(This is a separate DSECT for the RPL's RTNCD, FDBK, and FDBK2 fields.)
Session Parameters	(for DOS/VS and OS/VS)	ISTDBIND	(This is a separate DSECT which describes the session parameters obtained by INQUIRE or sent by OPNDST.)**
NIB	(for DOS/VS and OS/VS)	ISTDNIB	

Control Block		DSECT Name and Operands*	
NIB	(for DOS/VS and OS/VS)	ISTDVCHR***	(This is a separate DSECT for the NIB's DEVCHAR field.)
NIB	(for DOS/VS and OS/VS)	ISTDPROC***	(This is a separate DSECT for the NIB's PROC field.)

*This is what you code in your program to assemble the DSECT.

**This DSECT is described in Appendix J.

***If the DSECT for the entire NIB is used (ISTDNIB), the DSECT for this field is included automatically and should not be specified.

The format maps and the DSECT descriptions identify both the external field name (the declarative or manipulative macro keyword as used throughout this manual) and the internal field name (DSECT label) for each control block field. The DSECT descriptions are arranged in alphabetical order according to the external field name. To avoid the risk of duplicating DSECT labels, avoid using any label in your program that begins with the following characters: ACB, EXL, RPL, NIB, PRO, DEV, BIN, or USF. Users of these DSECTs must be very careful to set all relevant bits and fields. Mutually exclusive settings are indicated by indentations in the "Meaning" column. Related settings all appear under the same external name in the "Field" column.

If you compare listings of the actual DSECTs with the DSECT descriptions provided here, you will note that the actual DSECTs are more extensive. The fields that have been eliminated here are primarily fields that are set and used by VTAM, not by the application program. The control block fields that you set or examine should be limited to those fields that are included in the DSECT descriptions in this manual. (For this reason, you should not use a DSECT to initialize a control block; use GENCB or the appropriate ACB, EXLST, RPL, or NIB macro instruction instead.)

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label for EQU
ACBABNDP	ACB	ERROR	EQU	ACBERFLG
ACBAM	ACB	AM	DS or ORG	
ACBAPID	ACB	APPLID	DS or ORG	
ACBCALR	ACB	ERROR	EQU	ACBERFLG
ACBCDSNR	ACB	ERROR	EQU	ACBERFLG
ACBERFLG	ACB	ERROR	DS or ORG	
ACBLENG	ACB	ACBLEN	DS or ORG	
ACBLOGON	ACB	MACRF	EQU	ACBMACR2
ACBMACR2	ACB	MACRF	DS or ORG	
ACBOAHLT	ACB	ERROR	EQU	ACBERFLG
ACBOALR	ACB	ERROR	EQU	ACBERFLG
ACBOANAT	ACB	ERROR	EQU	ACBERFLG
ACBOANSN	ACB	ERROR	EQU	ACBERFLG
ACBOAPAA	ACB	ERROR	EQU	ACBERFLG
ACBOAPLE	ACB	ERROR	EQU	ACBERFLG
ACBOAPNM	ACB	ERROR	EQU	ACBERFLG
ACBOAVFY	ACB	ERROR	EQU	ACBERFLG
ACBOFLGS	ACB	OFLAGS	DS or ORG	
ACBONVRT	ACB	ERROR	EQU	ACBERFLG
ACBOPEN	ACB	OFLAGS	EQU	ACBOFLGS
ACBOPWLE	ACB	ERROR	EQU	ACBERFLG
ACBOPWSE	ACB	ERROR	EQU	ACBERFLG
ACBOUNDF	ACB	ERROR	EQU	ACBERFLG
ACBOVINA	ACB	ERROR	EQU	ACBERFLG
ACBPASSW	ACB	PASSWD	DS or ORG	
ACBUEL	ACB	EXLST	DS or ORG	
ACBVTAM	ACB	AM	EQU	ACBAM
DEVCATTN	NIB	DEVCHAR	EQU	DEVFLAGS
DEVCBSC	NIB	DEVCHAR	EQU	DEVFLAGS
DEVCC	NIB	DEVCHAR	EQU	DEVTCODE
DEVCCHEK	NIB	DEVCHAR	EQU	DEVFLAGS
DEVCCTL	NIB	DEVCHAR	EQU	DEVSHCH
DEVCHAR3	NIB	DEVCHAR	EQU	DEVFLAGS
DEVCODE	NIB	DEVCHAR		
DEVCONVR	NIB	DEVCHAR	EQU	DEVSHCH
DEVCRVB	NIB	DEVCHAR	EQU	DEVFLAGS
DEVCSLPN	NIB	DEVCHAR	EQU	DEVFLAGS
DEVCSSL	NIB	DEVCHAR	EQU	DEVFLAGS
DEVCSTCL	NIB	DEVCHAR	EQU	DEVFLAGS
DEVCSWL	NIB	DEVCHAR	EQU	DEVFLAGS
DEVFCCTL	NIB	DEVCHAR	EQU	DEVSHCH
DEVFLAGS	NIB	DEVCHAR	DS or ORG	
DEVINPUT	NIB	DEVCHAR	EQU	DEVSHCH
DEVLU	NIB	DEVCHAR	EQU	DEVTCODE
DEVMCODE	NIB	DEVCHAR	DS or ORG	
DEVMOD1	NIB	DEVCHAR	EQU	DEVMCODE
DEVMOD2	NIB	DEVCHAR	EQU	DEVMCODE
DEVMTA	NIB	DEVCHAR	EQU	DEVTCODE
DEVNNSPT	NIB	DEVCHAR	EQU	DEVSHCH
DEVOTPUT	NIB	DEVCHAR	EQU	DEVSHCH
DEVPHYSA	NIB	DEVCHAR	DS or ORG	

Figure H-1 (Part 1 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label for EQU
DEVRSV01	NIB	DEVCHAR	EQU	DEVSHCH
DEVSHCH	NIB	DEVCHAR	DS or ORG	
DEVSPS	NIB	DEVCHAR	EQU	DEVSHCH
DEVSUBND	NIB	DEVCHAR	EQU	DEVSHCH
DEVSYS3	NIB	DEVCHAR	EQU	DEVTCODE
DEVTCODE	NIB	DEVCHAR	DS or ORG	
DEVTWX	NIB	DEVCHAR	EQU	DEVTCODE
DEVWTTY	NIB	DEVCHAR	EQU	DEVTCODE
DEV50	NIB	DEVCHAR	EQU	DEVTCODE
DEV83B3	NIB	DEVCHAR	EQU	DEVTCODE
DEV115A	NIB	DEVCHAR	EQU	DEVTCODE
DEV545	NIB	DEVCHAR	EQU	DEVTCODE
DEV1017	NIB	DEVCHAR	EQU	DEVTCODE
DEV1018	NIB	DEVCHAR	EQU	DEVTCODE
DEV1050	NIB	DEVCHAR	EQU	DEVTCODE
DEV1052	NIB	DEVCHAR	EQU	DEVTCODE
DEV1053	NIB	DEVCHAR	EQU	DEVTCODE
DEV1054	NIB	DEVCHAR	EQU	DEVTCODE
DEV1055	NIB	DEVCHAR	EQU	DEVTCODE
DEV1056	NIB	DEVCHAR	EQU	DEVTCODE
DEV1057	NIB	DEVCHAR	EQU	DEVTCODE
DEV1058	NIB	DEVCHAR	EQU	DEVTCODE
DEV1092	NIB	DEVCHAR	EQU	DEVTCODE
DEV1130	NIB	DEVCHAR	EQU	DEVTCODE
DEV1255	NIB	DEVCHAR	EQU	DEVTCODE
DEV1800	NIB	DEVCHAR	EQU	DEVTCODE
DEV2203	NIB	DEVCHAR	EQU	DEVTCODE
DEV2213	NIB	DEVCHAR	EQU	DEVTCODE
DEV2265	NIB	DEVCHAR	EQU	DEVTCODE
DEV2502	NIB	DEVCHAR	EQU	DEVTCODE
DEV2701	NIB	DEVCHAR	EQU	DEVTCODE
DEV2703	NIB	DEVCHAR	EQU	DEVTCODE
DEV2715	NIB	DEVCHAR	EQU	DEVTCODE
DEV2740	NIB	DEVCHAR	EQU	DEVTCODE
DEV2741	NIB	DEVCHAR	EQU	DEVTCODE
DEV2770	NIB	DEVCHAR	EQU	DEVTCODE
DEV2780	NIB	DEVCHAR	EQU	DEVTCODE
DEV2972	NIB	DEVCHAR	EQU	DEVTCODE
DEV2980	NIB	DEVCHAR	EQU	DEVTCODE
DEV3125	NIB	DEVCHAR	EQU	DEVTCODE
DEV3135	NIB	DEVCHAR	EQU	DEVTCODE
DEV3271	NIB	DEVCHAR	EQU	DEVTCODE
DEV3272	NIB	DEVCHAR	EQU	DEVTCODE
DEV3275	NIB	DEVCHAR	EQU	DEVTCODE
DEV3277	NIB	DEVCHAR	EQU	DEVTCODE
DEV3284	NIB	DEVCHAR	EQU	DEVTCODE
DEV3286	NIB	DEVCHAR	EQU	DEVTCODE
DEV3704	NIB	DEVCHAR	EQU	DEVTCODE
DEV3705	NIB	DEVCHAR	EQU	DEVTCODE
DEV3735	NIB	DEVCHAR	EQU	DEVTCODE
DEV3741	NIB	DEVCHAR	EQU	DEVTCODE
DEV3747	NIB	DEVCHAR	EQU	DEVTCODE
DEV3780	NIB	DEVCHAR	EQU	DEVTCODE

Figure H-1 (Part 2 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label for EQU
DEV5496	NIB	DEVCHAR	EQU	DEVTCODE
EXLATTNF	EXLST	ATTN	DS or ORG	
EXLATTNP	EXLST	ATTN	DS or ORG	
EXLATTNS	EXLST	ATTN	EQU	EXLATTNF
EXLDFASF	EXLST	DFASY	DS or ORG	
EXLDFASP	EXLST	DFASY	DS or ORG	
EXLDFASS	EXLST	DFASY	EQU	EXLDFASF
EXLLEN2	EXLST	EXLLEN	DS or ORG	
EXLLERF	EXLST	LERAD	DS or ORG	
EXLLERP	EXLST	LERAD	DS or ORG	
EXLLERS	EXLST	LERAD	EQU	EXLLERF
EXLLGNF	EXLST	LOGON	DS or ORG	
EXLLGNP	EXLST	LOGON	DS or ORG	
EXLLGNS	EXLST	LOGON	EQU	EXLLGNF
EXLNLGNF	EXLST	LOSTERM	DS or ORG	
EXLNLGNP	EXLST	LOSTERM	DS or ORG	
EXLNLGNS	EXLST	LOSTERM	EQU	EXLNLGNF
EXLRESPF	EXLST	RESP	DS or ORG	
EXLRESPP	EXLST	RESP	DS or ORG	
EXLRESPS	EXLST	RESP	EQU	EXLRESPF
EXLRLRQF	EXLST	RELREQ	DS or ORG	
EXLRLRQP	EXLST	RELREQ	DS or ORG	
EXLRLRQS	EXLST	RELREQ	EQ	EXLRLRQF
EXLSCIPF	EXLST	SCIP	DS or ORG	
EXLSCIPP	EXLST	SCIP	DS or ORG	
EXLSCIPS	EXLST	SCIP	EQU	EXLSCIPF
EXLSYNF	EXLST	SYNAD	DS or ORG	
EXLSYNP	EXLST	SYNAD	DS or ORG	
EXLSYNS	EXLST	SYNAD	EQU	EXLSYNF
EXLTPNDF	EXLST	TPEND	DS or ORG	
EXLTPNDP	EXLST	TPEND	DS or ORG	
EXLTPNDS	EXLST	TPEND	EQU	EXLTPNDF
NIBCID	NIB	CID	DS or ORG	
NIBCON	NIB	CON	EQU	NIBFLG1
NIBDEVCH	NIB	DEVCHAR	DS or ORG	
NIBEXLST	NIB	EXLST	DS or ORG	
NIBFLG1	NIB	CON, LISTEND, SDT	DS or ORG	
NIBLAST	NIB	LISTEND	EQU	NIBFLG1
NIBLEN	NIB	NIBLEN	DS or ORG	
NIBLIMIT	NIB	RESPLIM	DS or ORG	
NIBLMODE	NIB	LOGMODE	DS or ORG	
NIBMODE	NIB	MODE	DS or ORG	
NIBNDAR	NIB	BNDAREA	DS or ORG	
NIBPROCD	NIB	PROC	DS or ORG	
NIBSDAPP	NIB	SDT	EQU	NIBFLG1
NIBSYM	NIB	NAME	DS or ORG	
NIBUSER	NIB	USERFLD	DS or ORG	
PROCA	NIB	PROC	EQU	PROPROC1
PROCFTX	NIB	PROC	EQU	PROPROC2
PROCS	NIB	PROC	EQU	PROPROC1

Figure H-1 (Part 3 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label or EQU
PRODFASY	NIB	PROC	EQU	PROPROC1
PROEIB	NIB	PROC	EQU	PROPROC4
PROEMLC	NIB	PROC	EQU	PROPROC2
PROERPI	NIB	PROC	EQU	PROPROC3
PROERPO	NIB	PROC	EQU	PROPROC2
PROLGIN	NIB	PROC	EQU	PROPROC3
PROLGOT	NIB	PROC	EQU	PROPROC2
PROMODB	NIB	PROC	EQU	PROPROC4
PROMODC	NIB	PROC	EQU	PROPROC4
PROMODM	NIB	PROC	EQU	PROPROC4
PROMODT	NIB	PROC	EQU	PROPROC4
PROMONIT	NIB	PROC	EQU	PROPROC3
PRONTFL	NIB	PROC	EQU	PROPROC2
PRONTO	NIB	PROC	EQU	PROPROC3
PROPROC1	NIB	PROC	DS or ORG	
PROPROC2	NIB	PROC	DS or ORG	
PROPROC3	NIB	PROC	DS or ORG	
PROPROC4	NIB	PROC	DS or ORG	
PRORESPX	NIB	PROC	EQU	PROPROC1
PRORPLC	NIB	PROC	EQU	PROPROC1
PROTRUNC	NIB	PROC	EQU	PROPROC1
PROXPOPT	NIB	PROC	EQU	PROPROC1
RPLAAREA	RPL	AAREA	DS or ORG	
RPLAARLN	RPL	AAREALN	DS or ORG	
RPLACTIV	RPL	CHECK	DS or ORG	
RPLAPPST	RPL	OPTCD	EQU	RPLOPT9
RPLARCLN	RPL	ARECLEN	DS or ORG	
RPLAREA	RPL	AREA	DS or ORG	
RPLARG	RPL	ARG-NIB	DS or ORG	
RPLASY	RPL	OPTCD	EQU	RPLOPT1
RPLBB	RPL	BRACKET	EQU	RPLRH3
RPLBID	RPL	CONTROL	EQU	RPLCNTCD
RPLBRANC	RPL	BRANCH	EQU	RPLEXTDS
RPLBSCID	RPL	OPTCD	EQU	RPLOPT10
RPLBUFL	RPL	AREALEN	DS or ORG	
RPLCBLKE	RPL	RTNCD	EQU	RPLRTNCD
RPLCHASE	RPL	CONTROL	EQU	RPLCNTDF
RPLCHN	RPL	CHAIN	DS or ORG	
RPLCHNG	RPL	REQ	EQU	RPLREQ
RPLCHREQ	RPL	CHNGDIR	EQU	RPLRH3
RPLCIDE	RPL	OPTCD	EQU	RPLOPT9
RPLCLACB	RPL	REQ	EQU	RPLREQ
RPLCLEAR	RPL	CONTROL	EQU	RPLCNTSC
RPLCLOSE	RPL	REQ	EQU	RPLREQ
RPLCMD	RPL	CHNGDIR	EQU	RPLRH3
RPLCMDRT	RPL	RTNCD	EQU	RPLRTNCD
RPLCNALL	RPL	OPTCD	EQU	RPLOPT7
RPLCNANY	RPL	OPTCD	EQU	RPLOPT7
RPLCNCEL	RPL	CONTROL	EQU	RPLCNTDF
RPLCNTDC	RPL	CONTROL	DS or ORG	
RPLCNTDF	RPL	CONTROL	DS or ORG	

Figure H-1 (Part 4 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label or EQU
RPLCNTSC	RPL	CONTROL	DS or ORG	
RPLCOND	RPL	OPTCD	EQU	RPLOPT6
RPLCOUNT	RPL	OPTCD	EQU	RPLOPT9
RPLCPMI	RPL	SSENSEI	EQU	RPLSSEI
RPLCPMO	RPL	SSENSEO	EQU	RPLSSEO
RPLCSI	RPL	CODESEL	EQU	RPLRH3
RPLDACB	RPL	ACB	DS or ORG	
RPLDATA	RPL	CONTROL	EQU	RPLCNTDF
RPLDEVCH	RPL	OPTCD	EQU	RPLOPT9
RPLDEVDC	RPL	RTNCD	EQU	RPLRTNCD
RPLDFASY	RPL	RTYPE	EQU	RPLSRTYP
RPLDLGIN	RPL	OPTCD	EQU	RPLOPT5
RPLDO	RPL	REQ	EQU	RPLREQ
RPLDSPLY	RPL	OPTCD	EQU	RPLOPT10
RPLEAU	RPL	OPTCD	EQU	RPLOPT5
RPLEB	RPL	BRACKET	EQU	RPLRH3
RPLECB	RPL	CHECK, ECB-EXIT	DS or ORG	
RPLECBIN	RPL	ECB	EQU	RPLOPT1
RPLEOB	RPL	OPTCD	EQU	RPLOPT6
RPLEOM	RPL	OPTCD	EQU	RPLOPT6
RPLEOT	RPL	OPTCD	EQU	RPLOPT6
RPLERASE	RPL	OPTCD	EQU	RPLOPT5
RPLEX	RPL	RESPOND	EQU	RPLVTFL2
RPLEXIT	RPL	EXIT	EQU	RPLEXTDS
RPLEXRQ	RPL	RTNCD	EQU	RPLRTNCD
RPLEXRS	RPL	RTNCD	EQU	RPLRTNCD
RPLEXSCH	RPL	CHECK	EQU	RPLEXTDS
RPLEXTDS	RPL	ARG-NIB, BRANCH, CHECK, EXIT	DS or ORG	
RPLFDB2	RPL	FDBK2	DS or ORG	
RPLFDB3	RPL	FDBK-DATAFLG	DS or ORG	
RPLFDBK2	RPL	SENSE	DS or ORG	
RPLFII	RPL	SSENSEI	EQU	RPLSSEI
RPLFIO	RPL	SSENSEO	EQU	RPLSSEO
RPLFIRST	RPL	CHAIN	EQU	RPLCHN
RPLFMHDR	RPL	OPTCD	EQU	RPLOPT12
RPLIBSQ	RPL	IBSQAC	DS or ORG	
RPLIBSQV	RPL	IBSQVAL	DS or ORG	
RPLIIGN	RPL	IBSQAC	EQU	RPLIBSQ
RPLIINV	RPL	IBSQAC	EQU	RPLIBSQ
RPLINEG	RPL	IBSQAC	EQU	RPLIBSQ
RPLINQIR	RPL	REQ	EQU	RPLREQ
RPLINTPT	RPL	REQ	EQU	RPLREQ
RPLIPOS	RPL	IBSQAC	EQU	RPLIBSQ
RPLIRSET	RPL	IBSQAC	EQU	RPLIBSQ
RPLISET	RPL	IBSQAC	EQU	RPLIBSQ
RPLITST	RPL	IBSQAC	EQU	RPLIBSQ
RPLKEEP	RPL	OPTCD	EQU	RPLOPT12

Figure H-1 (Part 5 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label or EQU
RPLLAST	RPL	CHAIN	EQU	RPLCHN
RPLLEN2	RPL	RPLLEN	DS or ORG	
RPLLIMEX	RPL	RTNCD	EQU	RPLRTNCD
RPLLOCK	RPL	OPTCD	EQU	RPLOPT6
RPLLOGIC	RPL	RTNCD	EQU	RPLRTNCD
RPLLOGON	RPL	OPTCD	EQU	RPLOPT9
RPLLUS	RPL	CONTROL	EQU	RPLCNTDC
RPLMIDLE	RPL	CHAIN	EQU	RPLCHN
RPLNCOND	RPL	OPTCD	EQU	RPLOPT6
RPLNERAS	RPL	OPTCD	EQU	RPLOPT5
RPLNEXIT	RPL	EXIT	EQU	RPLEXTDS
RPLNFME	RPL	RESPOND	EQU	RPLVTFL2
RPLNFSYN	RPL	RTYPE	EQU	RPLSRTYP
RPLNGRCC	RPL	RTNCD	EQU	RPLRTNCD
RPLNIB	RPL	ARG-NIB	EQU	RPLEXTDS
RPLNIBTK	RPL	OPTCD	EQU	RPLOPT12
RPLNODE	RPL	OPTCD	EQU	RPLOPT5
RPLNOERR	RPL	RTNCD	EQU	RPLRTNCD
RPLNOIN	RPL	RTNCD	EQU	RPLRTNCD
RPLOBSQ	RPL	OBSQAC	DS or ORG	
RPLOBSQV	RPL	OBSQVAL	DS or ORG	
RPLODACP	RPL	OPTCD	EQU	RPLOPT8
RPLODACQ	RPL	OPTCD	EQU	RPLOPT8
RPLOIGN	RPL	OBSQAC	EQU	RPLOBSQ
RPLOINV	RPL	OBSQAC	EQU	RPLOBSQ
RPLONEG	RPL	OBSQAC	EQU	RPLOBSQ
RPLONLY	RPL	CHAIN	EQU	RPLCHN
RPLOPNDS	RPL	REQ	EQU	RPLREQ
RPLOPOS	RPL	OBSQAC	EQU	RPLOBSQ
RPLOPT1	RPL	ECB, OPTCD	DS or ORG	
RPLOPT5	RPL	OPTCD	DS or ORG	
RPLOPT6	RPL	OPTCD	DS or ORG	
RPLOPT7	RPL	OPTCD	DS or ORG	
RPLOPT8	RPL	OPTCD	DS or ORG	
RPLOPT9	RPL	OPTCD	DS or ORG	
RPLOPT10	RPL	OPTCD	DS or ORG	
RPLOPT11	RPL	OPTCD	DS or ORG	
RPLOPT12	RPL	OPTCD	DS or ORG	
RPLORSET	RPL	OBSQAC	EQU	RPLOBSQ
RPLOSET	RPL	OBSQAC	EQU	RPLOBSQ
RPLOTST	RPL	OBSQAC	EQU	RPLOBSQ
RPLPATHI	RPL	SSENSEI	EQU	RPLSSEI
RPLPHYSC	RPL	RTNCD	EQU	RPLRTNCD
RPLPOST	RPL	CHECK	EQU	RPLECB
RPLPSOPT	RPL	OPTCD	EQU	RPLOPT5
RPLPURGE	RPL	RTNCD	EQU	RPLRTNCD
RPLQC	RPL	CONTROL	EQU	RPLCNTDF
RPLQEC	RPL	CONTROL	EQU	RPLCNTDF
RPLQOPT	RPL	OPTCD	EQU	RPLOPT7
RPLQUIES	RPL	OPTCD	EQU	RPLOPT11
RPLQUISE	RPL	REQ	EQU	RPLREQ

Figure H-1 (Part 6 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label or EQU
RPLRCVCD	RPL	REQ	EQU	RPLREQ
RPLRDSOH	RPL	FDBK-DATAFLG	EQU	RPLFDB3
RPLREAD	RPL	REQ	EQU	RPLREQ
RPLRELQ	RPL	CONTROL	EQU	RPLCNTDF
RPLREOB	RPL	FDBK-DATAFLG	EQU	RPLFDB3
RPLREOM	RPL	FDBK-DATAFLG	EQU	RPLFDB3
RPLREOT	RPL	FDBK-DATAFLG	EQU	RPLFDB3
RPLREQ	RPL	REQ	DS or ORG	
RPLRESET	RPL	REQ	EQU	RPLREQ
RPLRH3	RPL	BRACKET, CHNDIR, CODESEL	DS or ORG	
RPLRLEN	RPL	RECLEN	DS or ORG	
RPLRLG	RPL	FDBK-DATAFLG	EQU	RPLFDB3
RPLRQR	RPL	CONTROL	EQU	RPLCNTSC
RPLRRESP	RPL	RTYPE	EQU	RPLSRTYP
RPLRRI	RPL	SSENSEI	EQU	RPLSSEI
RPLRRN	RPL	RESPOND	EQU	RPLVTFL2
RPLRRO	RPL	SSENSEO	EQU	RPLSSEO
RPLRSHUT	RPL	CONTROL	EQU	RPLCNTSC
RPLRSRCD	RPL	REQ	EQU	RPLREQ
RPLRTNCD	RPL	RTNCD	DS or ORG	
RPLRTR	RPL	CONTROL	EQU	RPLCNTDC
RPLSAV13	RPL		DS or ORG	
RPLSCHED	RPL	POST	EQU	RPLVTFL2
RPLSDT	RPL	CONTROL	EQU	RPLCNTSC
RPLSEQNO	RPL	SEQNO	DS or ORG	
RPLSHUTC	RPL	CONTROL	EQU	RPLCNTSC
RPLSHUTD	RPL	CONTROL	EQU	RPLCNTSC
RPLSIGDA	RPL	SIGDATA	DS or ORG	
RPLSIGNL	RPL	CONTROL	EQU	RPLCNTDC
RPLSLICT	RPL	REQ	EQU	RPLREQ
RPLSMLGO	RPL	REQ	EQU	RPLREQ
RPLSNDCD	RPL	REQ	EQU	RPLREQ
RPLSPECC	RPL	RTNCD	EQU	RPLRTNCD
RPLSRESP	RPL	STYPE	EQU	RPLSRTYP
RPLSRTYP	RPL	RTYPE, STYPE	DS or ORG	
RPLSSCCD	RPL	REQ	EQU	RPLREQ
RPLSSEI	RPL	SSENSEI	DS or ORG	
RPLSSEO	RPL	SSENSEO	DS or ORG	
RPLSSMI	RPL	SSENSMI	DS or ORG	
RPLSSMO	RPL	SSENSMO	DS or ORG	
RPLSTART	RPL	OPTCD	EQU	RPLOPT11
RPLSTATI	RPL	SSENSEI	EQU	RPLSSEI
RPLSTATO	RPL	SSENSEO	EQU	RPLSSEO
RPLSTOP	RPL	OPTCD	EQU	RPLOPT11
RPLSTSN	RPL	CONTROL	EQU	RPLCNTSC
RPLSYERR	RPL	RTNCD	EQU	RPLRTNCD
RPLTERMS	RPL	OPTCD	EQU	RPLOPT9
RPLTOPL	RPL	OPTCD	EQU	RPLOPT9
RPLTRUNC	RPL	OPTCD	EQU	RPLOPT12

Figure H-1 (Part 7 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label or EQU
RPLUINPT	RPL	FDBK-DATAFLG	EQU	RPLFDB3
RPLUSFLD	RPL	USER	DS or ORG	
RPLUSNSI	RPL	USENSEI	DS or ORG	
RPLUSNSO	RPL	USENSEO	DS or ORG	
RPLVTFL2	RPL	POST, RESPOND	DS or ORG	
RPLVTMNA	RPL	RTNCD	EQU	RPLRTNCD
RPLWRITE	RPL	REQ	EQU	RPLREQ
RPLWROPT	RPL	OPTCD	EQU	RPLOPT5
USFABNDO	RPL	FDBK2	EQU	RPLFDB2
USFACINV	RPL	FDBK2	EQU	RPLFDB2
USFANS	RPL	FDBK2	EQU	RPLFDB2
USFAOK	RPL	FDBK2	EQU	RPLFDB2
USFAOOK	RPL	FDBK2	EQU	RPLFDB2
USFAPNAC	RPL	FDBK2	EQU	RPLFDB2
USFATNRC	RPL	FDBK2	EQU	RPLFDB2
USFATSFI	RPL	FDBK2	EQU	RPLFDB2
USFBHSUN	RPL	FDBK2	EQU	RPLFDB2
USFBSCSM	RPL	FDBK2	EQU	RPLFDB2
USFBTEOR	RPL	FDBK2	EQU	RPLFDB2
USFBTHEX	RPL	FDBK2	EQU	RPLFDB2
USFCBERR	RPL	FDBK2	EQU	RPLFDB2
USFCCCPY	RPL	FDBK2	EQU	RPLFDB2
USFCIDNG	RPL	FDBK2	EQU	RPLFDB2
USFCLOCC	RPL	FDBK2	EQU	RPLFDB2
USFCLRED	RPL	FDBK2	EQU	RPLFDB2
USFCLSIP	RPL	FDBK2	EQU	RPLFDB2
USFCPCNT	RPL	FDBK2	EQU	RPLFDB2
USFCPYE1	RPL	FDBK2	EQU	RPLFDB2
USFCPYE2	RPL	FDBK2	EQU	RPLFDB2
USFCRIRT	RPL	FDBK2	EQU	RPLFDB2
USFCRNF	RPL	FDBK2	EQU	RPLFDB2
USFCRPLN	RPL	FDBK2	EQU	RPLFDB2
USFCRSDC	RPL	FDBK2	EQU	RPLFDB2
USFCTERM	RPL	FDBK2	EQU	RPLFDB2
USFCTN32	RPL	FDBK2	EQU	RPLFDB2
USFCWB	RPL	FDBK2	EQU	RPLFDB2
USFCWTOO	RPL	FDBK2	EQU	RPLFDB2
USFDAMGE	RPL	FDBK2	EQU	RPLFDB2
USFDFIBH	RPL	FDBK2	EQU	RPLFDB2
USFDFIPO	RPL	FDBK2	EQU	RPLFDB2
USFDIDIL	RPL	FDBK2	EQU	RPLFDB2
USFDIDOL	RPL	FDBK2	EQU	RPLFDB2
USFDISCO	RPL	FDBK2	EQU	RPLFDB2
USFDSTIU	RPL	FDBK2	EQU	RPLFDB2
USFDSTNO	RPL	FDBK2	EQU	RPLFDB2
USFDSTUO	RPL	FDBK2	EQU	RPLFDB2
USFDUPND	RPL	FDBK2	EQU	RPLFDB2
USFDVUNS	RPL	FDBK2	EQU	RPLFDB2
USFENVER	RPL	FDBK2	EQU	RPLFDB2
USFEWAU3	RPL	FDBK2	EQU	RPLFDB2
USFEWBLK	RPL	FDBK2	EQU	RPLFDB2
USFEWNS	RPL	FDBK2	EQU	RPLFDB2

Figure H-1 (Part 8 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label or EQU
USFEXRQ	RPL	FDBK2	EQU	RPLFDB2
USFEXRS	RPL	FDBK2	EQU	RPLFDB2
USFEXTAZ	RPL	FDBK2	EQU	RPLFDB2
USFEXTEZ	RPL	FDBK2	EQU	RPLFDB2
USFIACT	RPL	FDBK2	EQU	RPLFDB2
USFICNDN	RPL	FDBK2	EQU	RPLFDB2
USFIDA	RPL	FDBK2	EQU	RPLFDB2
USFIDAEL	RPL	FDBK2	EQU	RPLFDB2
USFIICBE	RPL	FDBK2	EQU	RPLFDB2
USFIIINA	RPL	FDBK2	EQU	RPLFDB2
USFIINA	RPL	FDBK2	EQU	RPLFDB2
USFILDOA	RPL	FDBK2	EQU	RPLFDB2
USFILDOP	RPL	FDBK2	EQU	RPLFDB2
USFILNBL	RPL	FDBK2	EQU	RPLFDB2
USFILRS	RPL	FDBK2	EQU	RPLFDB2
USFILSIN	RPL	FDBK2	EQU	RPLFDB2
USFINA	RPL	FDBK2	EQU	RPLFDB2
USFINTNA	RPL	FDBK2	EQU	RPLFDB2
USFINVAP	RPL	FDBK2	EQU	RPLFDB2
USFINVLA	RPL	FDBK2	EQU	RPLFDB2
USFINVMD	RPL	FDBK2	EQU	RPLFDB2
USFINVNB	RPL	FDBK2	EQU	RPLFDB2
USFINVOT	RPL	FDBK2	EQU	RPLFDB2
USFINVRM	RPL	FDBK2	EQU	RPLFDB2
USFINVRT	RPL	FDBK2	EQU	RPLFDB2
USFINVSL	RPL	FDBK2	EQU	RPLFDB2
USFIOEDU	RPL	FDBK2	EQU	RPLFDB2
USFIQUIE	RPL	FDBK2	EQU	RPLFDB2
USFIREST	RPL	FDBK2	EQU	RPLFDB2
USFITNA	RPL	FDBK2	EQU	RPLFDB2
USFJTOT	RPL	FDBK2	EQU	RPLFDB2
USFLGCNT	RPL	FDBK2	EQU	RPLFDB2
USFLIMEX	RPL	FDBK2	EQU	RPLFDB2
USFLIORP	RPL	FDBK2	EQU	RPLFDB2
USFMBHSS	RPL	FDBK2	EQU	RPLFDB2
USFMCNVD	RPL	FDBK2	EQU	RPLFDB2
USFMDINC	RPL	FDBK2	EQU	RPLFDB2
USFMDNAU	RPL	FDBK2	EQU	RPLFDB2
USFMT100	RPL	FDBK2	EQU	RPLFDB2
USFNCPAO	RPL	FDBK2	EQU	RPLFDB2
USFNLGFA	RPL	FDBK2	EQU	RPLFDB2
USFNOIN	RPL	FDBK2	EQU	RPLFDB2
USFNONVR	RPL	FDBK2	EQU	RPLFDB2
USFNOPAU	RPL	FDBK2	EQU	RPLFDB2
USFNORD	RPL	FDBK2	EQU	RPLFDB2
USFNOTAS	RPL	FDBK2	EQU	RPLFDB2
USFNPSAU	RPL	FDBK2	EQU	RPLFDB2
USFOSDTF	RPL	FDBK2	EQU	RPLFDB2
USFPCF	RPL	FDBK2	EQU	RPLFDB2
USFPREXC	RPL	FDBK2	EQU	RPLFDB2
USFQSCIE	RPL	FDBK2	EQU	RPLFDB2
USFRCDPR	RPL	FDBK2	EQU	RPLFDB2

Figure H-1 (Part 9 of 10). Alphabetical List of Control Block Labels

DSECT Label	Control Block	Field Name	Type of Label	DS or ORG Label or EQU
USFRCINV	RPL	FDBK2	EQU	RPLFDB2
USFRCWNP	RPL	FDBK2	EQU	RPLFDB2
USFRECIP	RPL	FDBK2	EQU	RPLFDB2
USFRELNP	RPL	FDBK2	EQU	RPLFDB2
USFRESSU	RPL	FDBK2	EQU	RPLFDB2
USFREXAL	RPL	FDBK2	EQU	RPLFDB2
USFRILCP	RPL	FDBK2	EQU	RPLFDB2
USFRIOCC	RPL	FDBK2	EQU	RPLFDB2
USFRLGIC	RPL	FDBK2	EQU	RPLFDB2
USFRMD32	RPL	FDBK2	EQU	RPLFDB2
USFRNFT3	RPL	FDBK2	EQU	RPLFDB2
USFRNORT	RPL	FDBK2	EQU	RPLFDB2
USFRSCNC	RPL	FDBK2	EQU	RPLFDB2
USFRSCNO	RPL	FDBK2	EQU	RPLFDB2
USFRTOOD	RPL	FDBK2	EQU	RPLFDB2
USFRTRAF	RPL	FDBK2	EQU	RPLFDB2
USFRVIRC	RPL	FDBK2	EQU	RPLFDB2
USFSBFAL	RPL	FDBK2	EQU	RPLFDB2
USFSCEF	RPL	FDBK2	EQU	RPLFDB2
USFSCEM	RPL	FDBK2	EQU	RPLFDB2
USFSDFR	RPL	FDBK2	EQU	RPLFDB2
USFSDNP	RPL	FDBK2	EQU	RPLFDB2
USFSINVC	RPL	FDBK2	EQU	RPLFDB2
USFSINVR	RPL	FDBK2	EQU	RPLFDB2
USFSINVS	RPL	FDBK2	EQU	RPLFDB2
USFSNOS	RPL	FDBK2	EQU	RPLFDB2
USFSNOUT	RPL	FDBK2	EQU	RPLFDB2
USFSNQC	RPL	FDBK2	EQU	RPLFDB2
USFSSEQ	RPL	FDBK2	EQU	RPLFDB2
USFSTALF	RPL	FDBK2	EQU	RPLFDB2
USFSTOOD	RPL	FDBK2	EQU	RPLFDB2
USFSYERR	RPL	FDBK2	EQU	RPLFDB2
USFSYMNU	RPL	FDBK2	EQU	RPLFDB2
USFTANAV	RPL	FDBK2	EQU	RPLFDB2
USFTAPUA	RPL	FDBK2	EQU	RPLFDB2
USFUNTRM	RPL	FDBK2	EQU	RPLFDB2
USFUSELE	RPL	FDBK2	EQU	RPLFDB2
USFUSRES	RPL	FDBK2	EQU	RPLFDB2
USFUTSRC	RPL	FDBK2	EQU	RPLFDB2
USFVOFOC	RPL	FDBK2	EQU	RPLFDB2
USFVTBFO	RPL	FDBK2	EQU	RPLFDB2
USFVTHAL	RPL	FDBK2	EQU	RPLFDB2
USFVTMNA	RPL	FDBK2	EQU	RPLFDB2
USFWANCR	RPL	FDBK2	EQU	RPLFDB2
USFWCNVR	RPL	FDBK2	EQU	RPLFDB2
USFWTOI	RPL	FDBK2	EQU	RPLFDB2
USFXORDC	RPL	FDBK2	EQU	RPLFDB2
USFYTCTN	RPL	FDBK2	EQU	RPLFDB2
USFYTCTL	RPL	FDBK2	EQU	RPLFDB2

Figure H-1 (Part 10 of 10). Alphabetical List of Control Block Labels

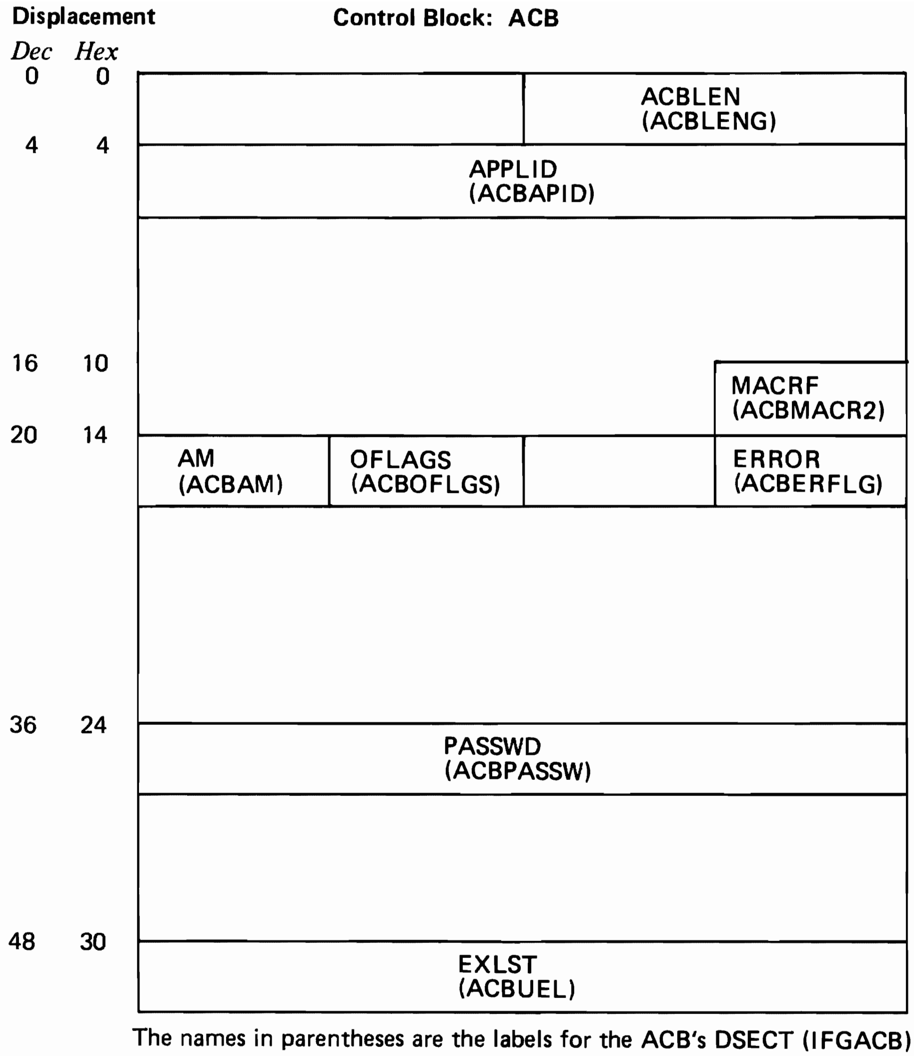
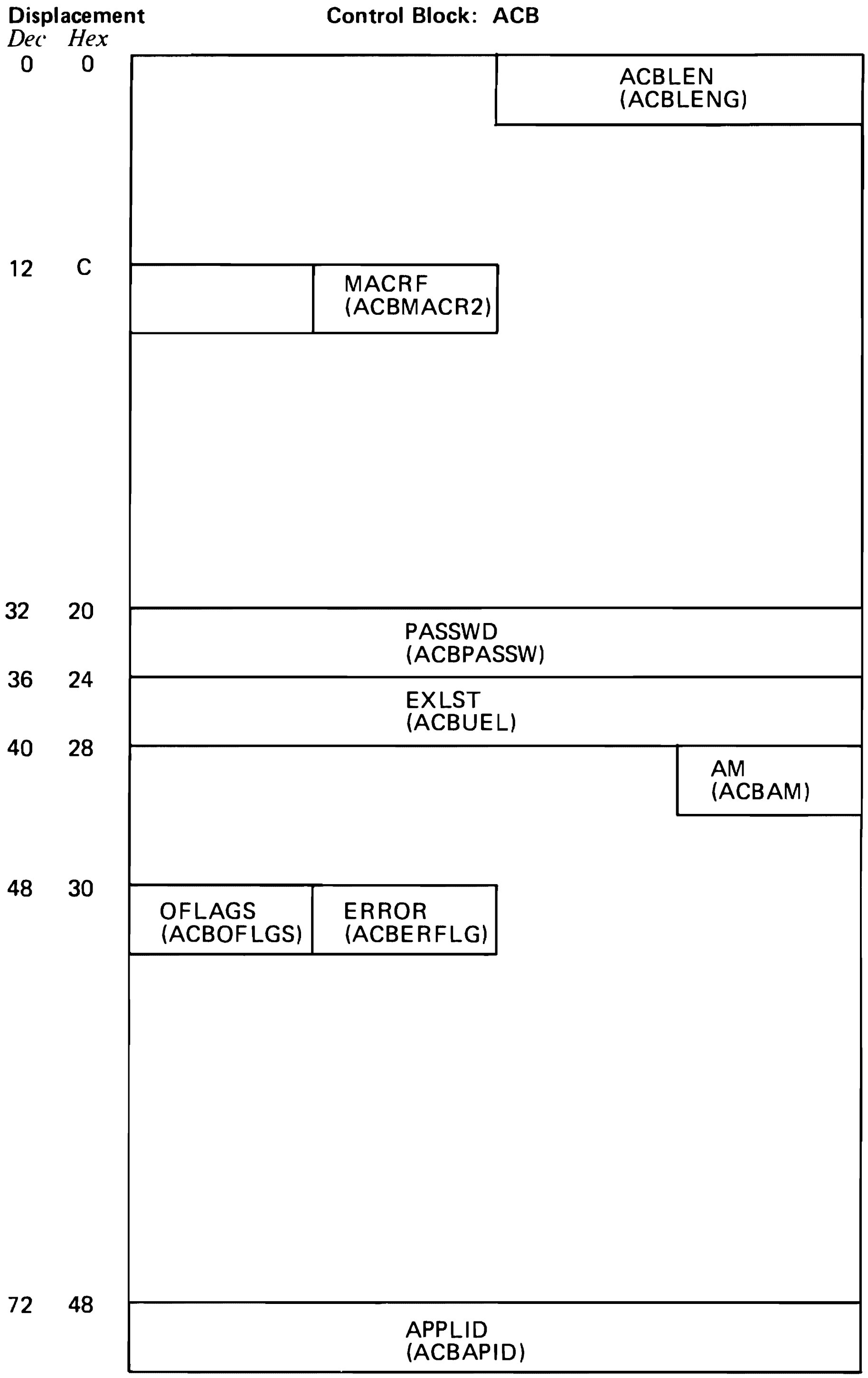


Figure H-2. The Format of the DOS/VS ACB



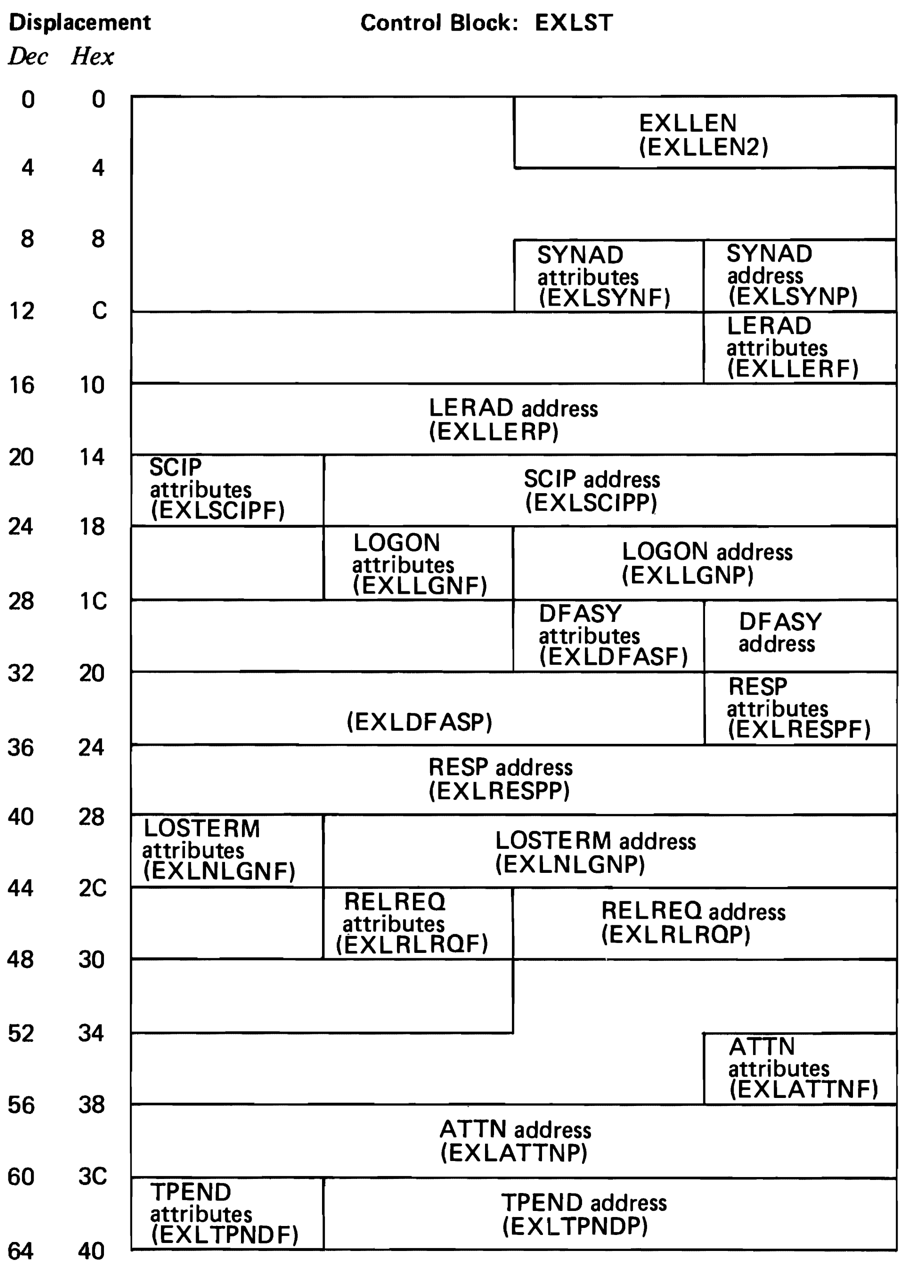
The names in parentheses are the labels for the ACB's DSECT (IFGACB)

Figure H-3. The Format of the OS/VS ACB

ACB DSECT: IFGACB

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
ACBLEN	ACBLENG	–	–	ACB length	2	2	2	2
AM	ACBAM	ACBVTAM	X'60'	AM=VTAM (after OPEN)	20	14	43	2B
APPLID	ACBAPID	–	–	APPLID address	4	4	72	48
ERROR	ACBERFLG	ACBOALR	X'04'	Already open (OPEN)	23	17	49	31
		ACBCALR	X'04'	Already closed (CLOSE)				
		ACBONVP	X'14'	Not enough storage (OPEN) (OS/VS only)				
			X'24'	Incorrect password (OPEN)				
		ACBCDSNR	X'42'	Some connections not released (CLOSE)				
		ACBNMAIN	X'46'	OPEN/CLOSE not in mainline (DOS/VS only)				
		ACBRNORF	X'4C'	Messages are queued or VTAM is waiting for a reply (CLOSE)				
			X'48'	CLOSE not issued in VTAM region (OS/VS1 and OS VS2 SVS only)				
		ACBOANAT	X'50'	VTAM not defined during system generation				
		ACBOAHLT	X'52'	VTAM is halting (OPEN)				
		ACBOAVFY	X'54'	APPLID is invalid (OPEN)				
		ACBOANSN	X'56'	APPLID is name of non-APPL (OPEN)				
		ACBOAPAA	X'58'	APPL is already active (OPEN)				
		ACBOAPNM	X'5A'	No matching APPL found (OPEN)				
		ACBOVINA	X'5C'	VTAM in system but inactive				
		ACBOAPSE	X'5E'	APPLID not in requestor's space				
		ACBOUNDF	X'60'	Undefined system error				
		ACBOAPLE	X'62'	APPLID length too small				
		ACBOPWSE	X'64'	Password not in requestor's space				
		ACBOPWLE	X'66'	Password length invalid				
		ACBRNOOF	X'68'	Primary program already active (OPEN)				
		ACBABNDP	X'70'	CLOSE rejected; program is being abended				
		ACBTVTCL	X'70'	OPEN rejected; application program is being closed and is unavailable				
		ACBONVRT	X'88'	Not enough storage (OPEN)(DOS/VS only)				
		ACBOACT	X'BC'	ACB active				
EXLST	ACBUEL	–	–	EXLST address	48	30	36	24
MACRF	ACBMACR2	ACBLOGON	X'08'	MACRF=NLOGON	19	13	13	D
OFLAGS	ACBOFLGS	ACBOPEN	X'10'	OFLAGS=OPEN	21	15	48	30
PASSWD	ACBPASSW	–	–	Password value	36	24	32	20

Figure H-4. The DOS/VS and OS/VS ACB DSECT (IFGACB)



The names in parentheses are the labels for the EXLST's DSECT (IFGEXLST)

Figure H-5. The Format of the DOS/VS and OS/VS EXLST

EXLST DSECT: IFGEXLST

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
ATTN	EXLATTNF	EXLATTNS	X'80'	ATTN exit present	55	37	55	37
	EXLATTNP	–	–	ATTN exit address	56	38	56	38
DFASY	EXLDFASF	EXLDFASS	X'80'	DFASY exit present	30	1E	30	1E
	EXLDFASP	–	–	DFASY exit address	31	1F	31	1F
EXLLEN	EXLLEN2	–	–	EXLST length	2	2	2	2
LERAD	EXLLERF	EXLLERS	X'80'	LERAD exit present	15	F	15	F
	EXLLERP	–	–	LERAD exit address	16	10	16	10
LOGON	EXLLGNF	EXLLGNS	X'80'	LOGON exit present	25	19	25	19
	EXLLGNP	–	–	LOGON exit address	26	1A	26	1A
LOSTERM	EXLNLGNF	EXLNLGNS	X'80'	LOSTERM exit present	40	28	40	28
	EXLNLGNP	–	–	LOSTERM exit address	41	29	41	29
RELREQ	EXLRLRQF	EXLRLRQS	X'80'	RELREQ exit present	45	2D	45	2D
	EXLRLRQP	–	–	RELREQ exit address	46	2E	46	2E
RESP	EXLRESPF	EXLRESPS	X'80'	RESP exit present	35	23	35	23
	EXLRESPP	–	–	RESP exit address	36	24	36	24
SCIP	EXLSCIPF	EXLSCIPS	X'80'	SCIP exit present	20	14	20	14
	EXLSCIPP	–	–	SCIP exit address	21	15	21	15
SYNAD	EXLSYNF	EXLSYNS	X'80'	SYNAD exit present	10	A	10	A
	EXLSYNP	–	–	SYNAD exit address	11	B	11	B
TPEND	EXLTPNDF	EXLTPNDS	X'80'	TPEND exit present	60	3C	60	3C
	EXLTPNDP	–	–	TPEND exit address	61	3D	61	3D

Figure H-6. The DOS/VS and OS/VS EXLST DSECT (IFGEXLST)

Displacement **Control Block: RPL**

Dec	*Hex*	Byte 0	Byte 1	Byte 2	Byte 3	Notes
0	0				RPLLEN (RPLLEN2)	
4	4					
8	8	NIB-ARG (RPLARG)				
12	C	AREA (RPLAREA)				See ISTDBIND (Figure J-1 in Appendix J when pointing to a bind area)
16	10	RECLEN (RPLRLEN)				
20	14	AREALEN (RPLBUFL)				
24	18	ACB (RPLDACB)				
28	1C		REQ (RPLREQ)			
32	20	OPTCD (RPLOPT1)		EXIT attributes (RPLEXTDS)		
36	24		RTNCD (RPLRTNCD)	FDBK2 (RPLFDB2)	DATAFLG-FDBK (RPLFDB3)	See ISTUSFBC (Figure H-10)
40	28	AAREA (RPLAAREA)				
44	2C	ECB-EXIT (RPLECB)				
48	30	AAREALN (RPLAARLN)				
52	34	ARECLEN (RPLARCLN)				
56	38	SENSE (RPLFDBK2): SSENSEI (RPLSSEI)	SSENSMI (RPLSSMI)	USENSEI (RPLUSNSI)		
60	3C	USER (RPLUSFLD)				

Figure H-7 (Part 1 of 2). The Format of the DOS/VS RPL

Displacement **Control Block: RPL (Cont.)**

Dec	*Hex*				
64	40	OPTCD (RPLOPT5)	OPTCD (RPLOPT6)	OPTCD (RPLOPT7)	OPTCD (RPLOPT8)
68	44	OPTCD (RPLOPT9)	OPTCD (RPLOPT10)	OPTCD (RPLOPT11)	OPTCD (RPLOPT12)
72	48	BRACKET-CHNGDIR-CODESEL (RPLRH3)	STYPE-RTYPE (RPLSRTYP)		POST-RESPOND (RPLVTFL2)
76	4C	CHAIN (RPLCHN)	CONTROL(1) (RPLCNTDF)	CONTROL(2) (RPLCNTDC)	CONTROL(3) (RPLCNTSC)
80	50	OBSQVAL (RPLOBSQV)		IBSQVAL (RPLIBSQV)	
84	54	OBSQAC (RPLOBSQ)	IBSQAC (RPLIBSQ)	SEQNO (RPLSEQNO)	
88	58	SSENSEO (RPLSSEO)	SSENSMO (RPLSSMO)	USENSEO (RPLUSNSO)	
92	5C	CHECK (RPLACTIV)			
96	60	SIGDATA (RPLSIGDA)			

The names in parentheses are the labels for the RPL's DSECT (IFGRPL)

Figure H-7 (Part 2 of 2). The Format of the DOS/VS RPL

Displacement **Control Block: RPL**

Dec	*Hex*				
0	0			REQ (RPLREQ)	RPLLEN (RPLLEN2)
4	4				
8	8	ECB-EXIT (RPLECB)			
12	C	See ISTUSFBC (Figure H-10)	(RPLFDBK)		
			RTNCD (RPLRTNCD)	FDBK2 (RPLFDB2)	DATAFLG-FDBK (RPLFDB3)
16	10	BRACKET-CHNGDIR-CODESEL (RPLRH3)	STYPE-RTYPE (RPLSRTYP)	CHAIN (RPLCHN)	
20	14	POST-RESPOND (RPLVTFL2)	CONTROL (RPLCNTRL)		
			(RPLCNTDF)	(RPLCHTDC)	(RPLCNTSC)
24	18	ACB (RPLDACB)			
28	1C				
32	20	AREA (RPLAREA)			
36	24	NIB-ARG (RPLARG)			
40	28	OPTCD (RPLOPTI)			
44	2C				
48	30	RECLEN (RPLRLEN)			
52	34	AREALEN (RPLBUFL)			
56	38	OPTCD (RPLOPT5)	OPTCD (RPLOPT6)	OPTCD (RPLOPT7)	OPTCD (RPLOPT8)
60	3C	OBSQVAL (RPLOBSQV)		IBSQVAL (RPLIBSQV)	
64	40	OBSQAC (RPLOBSQ)	IBSQAC (RPLIBSQ)	SEQNO (RPLSEQNO)	

Figure H-8 (Part 1 of 2). The Format of the OS/VS RPL

Displacement **Control Block: RPL (Cont.)**

Dec	Hex				
68	44	EXIT attributes (RPLEXTDS)	CHECK (RPLACTIV)		
72	48				
76	4C	AAREA (RPLAAREA)			
80	50	AAREALN (RPLAARLN)			
84	54	ARECLEN (RPLARCLN)			
88	58	← SENSE (RPLFDBK2) →			
		SSENSEI (RPLSSEI)	SSENSMI (RPLSSMI)	USENSEI (RPLUSNSI)	
92	5C	USER (RPLUSFLD)			
96	60	OPTCD (RPLOPT9)		OPTCD (RPLOPT11)	OPTCD (RPLOPT12)
100	64	← (RPLOSENS) →			
		SSENSEO (RPLSSEO)	SSENSMO (RPLSSMO)	USENSEO (RPLUSNSO)	
104	68	(RPLSAV13)			
108	6C	SIGDATA (RPLSIGDA)			

The names in parentheses are the labels for the RPL's DSECT(IFGRPL)

Figure H-8 (Part 2 of 2). The Format of the OS/VS RPL

RPL DSECT: IFGRPL

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
AAREA	RPLAAREA	–	–	AAREA address	40	28	76	4C
AAREALN	RPLAARLN	–	–	AAREALEN value	48	30	80	50
ACB	RPLDACB	–	–	ACB address	24	18	24	18
AREA	RPLAREA	–	–	AREA address	12	C	32	20
AREALEN	RPLBUFL	–	–	AREALEN value	20	14	52	34
ARECLEN	RPLARCLN	–	–	ARECLEN value	52	34	84	54
BRACKET	RPLRH3	RPLBB	X'80'	BRACKET=BB	72	48	16	10
		RPLEB	X'40'	BRACKET=EB				
BRANCH	RPLEXTDS	RPLBRANC	X'02'	BRANCH=YES	34	22	68	44
CHAIN	RPLCHN	RPLFIRST	X'80'	CHAIN=FIRST	76	4C	18	12
		RPLMIDLE	X'40'	=MIDDLE				
		RPLLAST	X'20'	=LAST				
		RPLONLY	X'10'	=ONLY				
CHECK	RPLEXTDS	RPLEXSCH	X'80'	RPL exit has been entered	34	22	68	44
	RPLACTIV	–	X'FF'	RPL is active	92	5C	69	45
	RPLECB	RPLPOST	X'40'	Internal ECB has been posted	–	–	8	8
	RPLECB+2	RPLPOST	X'80'	Internal ECB has been posted	46	2E	–	–
CHNGDIR	RPLRH3	RPLCMD	X'20'	CHNGDIR=CMD	72	48	16	10
		RPLCHREQ	X'10'	CHNGDIR=REQ				
CODESEL	RPLRH3	RPLCSI	X'08'	CODESEL=ALT	77	48	–	–
CONTROL	RPLCNTDF	RPLDATA	X'80'	CONTROL=DATA	77	4D	21	15
(all settings		RPLCNCEL	X'40'	=CANCEL				
mutually		RPLQC	X'20'	=QC				
exclusive)		RPLQEC	X'10'	=QEC				
		RPLCHASE	X'08'	=CHASE				
		RPLRELQ	X'04'	=RELQ				
	RPLCNTDC	RPLBID	X'80'	=BID	78	4E	22	16
		RPLRTR	X'40'	=RTR				
		RPLLUS	X'20'	=LUS				
		RPLSIGNL	X'10'	=SIGNAL				
	RPLCNTSC	RPLSDT	X'80'	=SDT	79	4F	23	17
		RPLCLEAR	X'40'	=CLEAR				
		RPLSTSN	X'20'	=STSN				
		RPLSHUTD	X'10'	=SHUTD				
		RPLSHUTC	X'08'	=SHUTC				
		RPLRQR	X'04'	=RQR				
		RPLRSHUT	X'02'	=RSHUTD				
ECB	RPLOPT1	RPLECBIN	X'01'	ECB is external to the RPL	32	20	40	28
ECB-EXIT	RPLECB	–	–	ECB, ECB address, or EXIT address	44	2C	8	8
EXIT	RPLEXTDS	RPLNEXIT	X'40'	No RPL exit specified	34	22	68	44
		RPLEXIT	X'20'	RPL exit specified				
FDBK2	RPLFDB2			FDBK2 return code (see Figure H-10)	38	26	14	E
FDBK-DATAFLG	RPLFDB3	RPLUINPT	X'80'	DATAFLG=UNSOL	39	27	15	F
		RPLREOB	X'20'	=EOB				
		RPLREOM	X'10'	=EOM				
		RPLREOT	X'08'	=EOT				
		RPLRLG	X'02'	=LG				
		RPLRDSOH	X'01'	=SOH				
IBSQAC	RPLIBSQ	RPLISET	X'80'	IBSQAC=SET	85	55	65	41
		RPLITST	X'40'	=TESTSET				
		RPLIRSET	X'20'	=RESET				
		RPLIIGN	X'10'	=IGNORE				
		RPLIPOS	X'08'	=TESTPOS				
		RPLINEG	X'04'	=TESTNEG				
		RPLIINV	X'02'	=INVALID				

Figure H-9 (Part 1 of 3). The DOS/VS and OS/VS RPL DSECT (IFGRPL)

Field	*DSECT DS or ORG label*	*DSECT EQU label*	*Value*	*Meaning (For EQU, meaning when bit setting is on or when byte value is set)*	*DOS/VS Displacement*		*OS/VS Displacement*	
					Dec	*Hex*	*Dec*	*Hex*
IBSQVAL	RPLIBSQV	–	–	IBSQVAL value	82	52	68	3E
NIB-ARG	RPLEXTDS	RPLNIB	X'04'	RPLARG points to a NIB	34	22	68	44
	RPLARG	–	–	NIB address or CID	8	8	36	24
OBSQAC	RPLOBSQ	RPLOSET	X'80'	OBSQAC=SET	84	54	64	40
		RPLOTST	X'40'	=TESTSET				
		RPLORSET	X'20'	=RESET				
		RPLOIGN	X'10'	=IGNORE				
		RPLOPOS	X'08'	=TESTPOS				
		RPLONEG	X'04'	=TESTNEG				
		RPLOINV	X'02'	=INVALID				
OBSQVAL	RPLOBSQV	–	–	OBSQVAL value	80	50	60	3C
OPTCD	RPLOPT1	RPLASY	X'08'	OPTCD=ASY	32	20	40	28
	RPLOPT5	RPLDLGIN	X'80'	OPTCD=CS	64	40	56	38
		RPLPSOPT	X'20'	OPTCD=PASS				
		RPLNERAS	X'10'	OPTCD=NERASE				
		RPLEAU	X'08'	=EAU				
		RPLERACE	X'04'	=ERASE				
		RPLNODE	X'02'	OPTCD=ANY				
		RPLWROPT	X'01'	OPTCD=CONV				
	RPLOPT6	RPLEOB	X'80'	OPTCD=BLK	65	41	57	39
		RPLEOM	X'40'	=LBM				
		RPLEOT	X'20'	=LBT				
		RPLCOND	X'10'	OPTCD=COND				
		RPLNCOND	X'08'	=UNCOND				
		RPLLOCK	X'04'	=LOCK				
	RPLOPT7	RPLCNALL	X'80'	OPTCD=CONALL	66	42	58	3A
		RPLCNANY	X'40'	=CONANY				
		RPLQOPT	X'10'	OPTCD=Q				
		RPLRLSOP	X'04'	OPTCD=RELRQ				
	RPLOPT8	RPLODACQ	X'80'	OPTCD=ACQUIRE	67	43	59	3B
		RPLODACP	X'40'	=ACCEPT				
	RPLOPT9	RPLLOGON	X'80'	OPTCD=LOGONMSG	68	44	96	60
		RPLDEVCH	X'40'	=DEVCHAR				
		RPLTERMS	X'20'	=TERMS				
		RPLCOUNT	X'10'	=COUNTS				
		RPLAPPST	X'08'	=APPSTAT				
		RPLCIDE	X'02'	=CIDXLATE				
		RPLTOPL	X'01'	=TOPLOGON				
	RPLOPT10	RPLBSCID	X'80'	OPTCD=BSCID	69	45		
		RPLDSPLY	X'40'	=DISPLAY				
	RPLOPT11	RPLQUIES	X'80'	OPTCD=QUIESCE	70	46	98	62
		RPLSTART	X'40'	=START				
		RPLSTOP	X'20'	=STOP				
	RPLOPT12	RPLKEEP	X'40'	OPTCD=KEEP	71	47	99	63
		RPLTRUNC	X'20'	=TRUNC				
		RPLNIBTK	X'10'	=NIBTK				
		RPLFMHDR	X'01'	=FMHDR				
POST	RPLVTFL2	RPLSCHED	X'80'	POST=SCHED	75	4B	20	14
RECLEN	RPLRLEN	–	–	RECLEN value	16	10	48	30
REQ	RPLREQ	RPLWRITE	X'11'	WRITE	29	1D	2	2
		RPLRESET	X'12'	RESET				
		RPLDO	X'13'	DO				
		RPLQUISE	X'15'	SETLOGON				
		RPLSMLGO	X'16'	SIMLOGON				
		RPLOPNDS	X'17'	OPNDST				
		RPLCHNG	X'19'	CHANGE				
		RPLINQIR	X'1A'	INQUIRE				

Figure H-9 (Part 2 of 3). The DOS/VS and OS/VS RPL DSECT (IFGRPL)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
REQ	RPLREQ	RPLINTPT	X'1B'	INTRPRET	29	1D	2	2
		RPLREAD	X'1D'	READ				
		RPLSLICT	X'1E'	SOLICIT				
		RPLCLOSE	X'1F'	CLSDST				
		RPLCLACB	X'21'	CLOSE				
		RPLSNDCD	X'22'	SEND				
		RPLRCVCD	X'23'	RECEIVE				
		RPLRSRCD	X'24'	RESETSR				
		RPLSSCCD	X'25'	SESSIONC				
		RPLSDCMD	X'27'	SENDCMD				
		RPLRVCMD	X'28'	RCVCMD				
RESPOND	RPLVTFL2	RPLEX	X'04'	RESPOND=EX	75	4B	20	14
		RPLNFME	X'02'	RESPOND=NFME				
		RPLRRN	X'01'	RESPOND=RRN				
RPLLEN	RPLLEN2	–	–	RPL length	3	3	3	3
RPLSAV13	RPLSAV13	–	–	Save area (OS/VS only)			104	68
RTNCD	RPLRTNCD	ISTUSFBC		(See Figure H-10)	37	25	13	D
RTYPE	RPLSRTYP	RPLRRESP	X'08'	RTYPE=RESP	73	49	17	11
		RPLNFSYN	X'04'	RTYPE=NDFSYN				
		RPLDFASY	X'02'	RTYPE=DFASY				
SENSE	RPLFDBK2	–	–	BASIC mode input sense (BSC S/S)	56	38	88	58
	RPLFDBK2+2	–	–	BTU response codes (2 bytes)	58	3A	90	5A
SEQNO	RPLSEQNO	–	–	SEQNO value	86	56	66	42
SIGDATA	RPLSIGDA	–	–	Signal data	96	60	108	60
SSENSEI	RPLSSEI	RPLPATHI	X'80'	SSENSEI=PATH	56	38	88	58
		RPLCPMI	X'40'	=CPM				
		RPLSTATI	X'20'	=STATE				
		RPLFII	X'10'	=FI				
		RPLRRI	X'08'	=RR				
SSENSEO	RPLSSEO	RPLCPMO	X'40'	SSENSEO=CPM	88	58	100	64
		RPLSTATO	X'20'	=STATE				
		RPLFIO	X'10'	=FI				
		RPLRRO	X'08'	=RR				
SSENSMI	RPLSSMI			System sense modifier value	57	39	89	59
SSENSMO	RPLSSMO			System sense modifier value (outgoing)	89	59	101	65
STYPE	RPLSRTYP	RPLSRESP	X'80'	STYPE=RESP	73	49	17	11
USENSEI	RPLUSNSI	–	–	USENSEI value	58	3A	90	5A
USENSEO	RPLUSNSO	–	–	USENSEO value	90	5A	102	66
USER	RPLUSFLD	–	–	USER value	60	3C	92	5C

Figure H-9 (Part 3 of 3). The DOS/VS and OS/VS RPL DSECT (IFGRPL)

RTNCD-FDBK-FDBK2 DSECT: ISTUSFBC

Field	*DSECT DS or ORG label*	*DSECT EQU label*	*Value*	*Meaning (For EQU, meaning when bit setting is on or when byte value is set)*	*DOS/VS Displacement Dec*	*DOS/VS Displacement Hex*	*OS/VS Displacement Dec*	*OS/VS Displacement Hex*
				The following byte values apply to the RTNCD field (RPLRTNCD in the IFGRPL DSECT):				
		USFAOK	X'00'	Normal or conditional completion				
		USFXORDC	X'04'	Extraordinary completion				
		USFRESSU	X'08'	Retriable-Reissue				
		USFDAMGE	X'0C'	Damage				
		USFENVER	X'10'	Environment error				
		USFLOGIC	X'14'	User logic error				
		USFRLGIC	X'18'	(Should not occur)				
				The following byte values apply to the FDBK2 field (RPLFDBK2 in the IFGRPL DSECT) when RTNCD is set to X'00':				
		USFAOOK	X'00'	Normal completion				
		USFRCWNP	X'01'	RESET (COND) issued with I/O in progress				
		USFRCDPR	X'02'	Normal completion with data				
		USFYTCTN	X'03'	Yielded to contention				
		USFYTCTL	X'04'	Yielded to contention, error lock set				
		USFATSFI	X'05'	Input area too small				
		USFNOIN	X'06'	No input available				
		USFIIINA	X'07'	INQUIRE information not available				
		USFDSTIU	X'08'	Terminal in use				
		USFNLGFA	X'09'	No logon requests				
				The following byte values apply when RTNCD is set to X'04':				
		USFRVIRC	X'00'	RVI received				
		USFATNRC	X'01'	Attention or reverse break received				
		USFBSCSM	X'02'	SENSE field set				
		USFEXRQ	X'03'	Exception condition for incoming message				
		USFEXRS	X'04'	Incoming response indicates exception condition				
				The following byte value applies when RTNCD is set to X'08':				
		USFSTALF	X'00'	Temporary storage shortage				
				The following byte values apply when RTNCD is set to X'0C':				
		USFIOEDU	X'00'	Error lock set				
		USFDVUNS	X'01'	Terminal not usable				
		USFUNTRM	X'02'	Request canceled by TRM				
		USFBTHEX	X'03'	Buffers now emptied				
		USFBTEOR	X'04'	Buffers filled				
		USFNCPAO	X'05'	NCP abended, restart successful				
		USFLIORP	X'06'	NCP abended, restart successful (Final I/O request)				
		USFRECIP	X'07'	Connection recovery in progress				
		USFRTRAF	X'08'	Logical unit restarted				
		USFQOPDC	X'09'	Queued OPNDST canceled by CLSDST				
		USFUSRES	X'0A'	Request canceled by RESET or RESETSR				

Figure H-10 (Part 1 of 4). The RPL's RTNCD-FDBK-FDBK2 DSECT (ISTUSFBC)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	Hex	OS/VS Displacement Dec	Hex
		USFCLOCC	X'0B'	Request canceled by CLSDST				
		USFCLRED	X'0C'	Request canceled by clear indicator				
		USFPREXC[1]	X'0D'	SEND canceled due to prior exception condition				
				The following byte values apply when RTNCD is set to X'10'				
		USFTANAV	X'00'	Terminal or APPL not available				
		USFSBFAL	X'01'	OPNDST failed for logical unit				
		USFTAPUA	X'02'	APPL does not accept logon requests				
		USFVTHAL	X'03'	HALT (quick) issued				
		USFILRS	X'04'	VTAM/NCP incompatibility				
		USFPCF	X'05'	Permanent channel failure				
		USFANS	X'06'	Automatic NCP shutdown				
		USFVOFOC	X'07'	Request canceled by VARY command				
		USFDISCO	X'08'	Dial-line disconnection				
		USFUTSCR	X'09'	Unconditional terminate self received				
		USFSYERR	X'0A'	VTAM error				
		USFDIDOL	X'0B'	Dial-out disconnection				
		USFDIDIL	X'0C'	Dial-in disconnection				
		USFVTMNA	X'0D'	VTAM inactive for APPL				
		USFABNDO	X'0E'	Abend for APPL's TCB				
		USFVTBFO	X'0F'	Buffers filled for record devices				
		USFCTERM	X'10'	COND Term-self				
		USFOSDTF	X'11'	SDT failure on OPNDST				
				The following byte values apply when RTNCD is set to X'14':				
		USFNONVR	X'00'	VSAM request				
		USFNOTAS	X'01'	Reserved				
		USFEXTAZ	X'02'	Zero EXIT field				
		USFEXTEZ	X'03'	Zero ECB field				
		USFCRPLN	X'04'	Inactive RPL checked				
		USFCBERR	X'10'	Control block invalid				
		USFRNORT	X'11'	RTYPE invalid for RECEIVE				
		USFCLSIP	X'12'	CLSDST in progress				
		USFCIDNG	X'13'	CID invalid				
		USFILDOP	X'14'	CMD field invalid				
		USFWANCR	X'15'	READ LDO not chained				
		USFSTOOD	X'16'	SOLICIT for output-only terminal				
		USFRTOOD	X'17'	READ for output-only terminal				
		USFWTOI	X'18'	WRITE for input-only terminal				
		USFEWNS	X'19'	WRITE ERASE for invalid terminal				
		USFEWAU3	X'1A'	WRITE EAU for invalid terminal				
		USFCWTOO	X'1B'	WRITE CONV for output-only terminal				
		USFCWB	X'1C'	WRITE ERASE and CONV				
		USFCCCPY	X'1D'	COPYLBM or COPYLBT chained				
		USFIDA	X'1E'	Invalid data or length				
		USFILDOA	X'1F'	LDO address invalid				
		USFJTOJ	X'20'	Reserved				
		USFMT100	X'21'	Over 100 LDOs				
		USFRILCP	X'22'	Reserved				
		USFCRIRT	X'23'	Request type invalid				
		USFRIOCC	X'24'	Invalid FLAGS for a READ LDO				
		USFEWBLK	X'25'	WRITE ERASE and BLK				
		USFCRSDC	X'26'	Reserved				
		USFIREST	X'27'	RESET option invalid				
		USFWBT32	X'28'	WRITE option invalid				

[1] This EQU label does not appear in DOS/VS Release 30 or OS/VS Release 3.0. Users must provide their own EQU statements.

Figure H-10 (Part 2 of 4). The RPL's RTNCD-FDBK-FDBK2 DSECT (ISTUSFBC)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
		USFRMD32	X'29'	READBUF for non-3270 terminal				
		USFCTN32	X'2A'	COPY operation to non-3270 terminal				
		USFWCNVR	X'2B'	WRITE CONV when data expected				
		USFRNFT3	X'2C'	Output not preceded by input				
		USFRCINV	X'2D'	RESET COND with error lock set				
		USFINVRM	X'2E'	BLOCK-MSG-TRANS-CONT invalid				
		USFLGCNT	X'2F'	Too many leading graphic characters				
		USFCPCNT	X'30'	Invalid COPYLBM or COPYLBT LEN				
		USFIDAEL	X'31'	Invalid data area				
		USFUSELE	X'32'	Request invalid for specified area				
		USFCRNF	X'33'	WRITE CONV reply not possible				
		USFNORD	X'34'	First I/O not READ or SOLICIT				
		USFCPYE2	X'35'	Terminals not on same control unit				
		USFRELNP	X'36'	RESET LOCK invalid				
		USFCPYE1	X'37'	Terminal not connected				
		USFDFIBH	X'38'	Reserved				
		USFDFIPO	X'39'	Invalid PROC option				
		USFQSCIE	X'3A'	Reserved				
		USFREXAL	X'3B'	NFME-NRRN response				
		USFSDNP	X'3C'	SEND SCHED still pending				
		USFSCEM	X'3D'	Reserved				
		USFSCEF	X'3E'	Reserved				
		USFSNQC	X'3F'	Reserved				
		USFSINVC	X'40'	CONTROL invalid				
		USFSDFR	X'41'	No SDT issued				
		USFSNOS	X'42'	Reserved				
		USFSNOUT	X'43'	Reserved				
		USFLIMEX	X'44'	RESPLIM exceeded				
		USFSSEQ	X'45'	Reserved				
		USFSINVS	X'46'	Reserved				
		USFSINVR	X'47'	Invalid SEND for 3270				
		USFINVRT	X'48'	Redundant clear indicator				
		USFACINV	X'49'	Invalid STSN indicator				
		USFICNDN	X'4A'	APPL name not available				
		USFILSIN	X'4B'	INTRPRET sequence invalid				
		USFIICBE	X'4C'	No terminal or APPL name				
		USFINTNA	X'4D'	No interpret table				
		USFILNBL	X'4E'	Invalid use of a NIB list				
		USFINVOT	X'4F'	ACQUIRE-ACCEPT invalid				
		USFINVAP	X'50'	CONANY-CONALL invalid				
		USFAPNAC	X'51'	APPL never accepts				
		USFINVNB	X'52'	NIB invalid				
		USFSYMNU	X'53'	Terminal or APPL name not found				
		USFDSTUO	X'54'	Invalid terminal name				
		USFNOPAU	X'55'	OPNDST ACQUIRE not authorized				
		USFMDINC	X'56'	Invalid MODE				
		USFINVMD	X'57'	No MODE				
		USFBHSUN	X'58'	Reserved				
		USFMDNAU	X'59'	Reserved				
		USFMBHSS	X'5A'	Reserved				
		USFINVLA	X'5B'	Invalid logon message address				
		USFDUPND	X'5C'	Duplicate terminal names				
		USFDSTNO	X'5D'	Terminal not connected				
		USFNPSAU	X'5E'	CLSDST PASS not authorized				
		USFRSCNO	X'5F'	CLSDST PASS invalid				

Figure H-10 (Part 3 of 4). The RPL's RTNCD-FDBK-FDBK2 DSECT (ISTUSFBC)

Field	*DSECT DS or ORG label*	*DSECT EQU label*	*Value*	*Meaning (For EQU, meaning when bit setting is on or when byte value is set)*	*DOS/VS Displacement Dec*	*DOS/VS Displacement Hex*	*OS/VS Displacement Dec*	*OS/VS Displacement Hex*
		USFRSCNC	X'60'	CLSDST RELEASE invalid				
		ISFINVSL	X'61'	SETLOGON invalid				
		USFMCNVD	X'62'	Invalid request for MODE				
		USFRNOEL	X'6C'	Exceeded limit on outstanding RCVEMD requests				
		USFRNONA	X'6D'	Application program not authorized				
		USFRNOSE	X'6E'	Syntax error in reply to rework operation message				
		USFRNOIA	X'6F'	SENDCMD/RCVCMD processor inactive				
		USFRNOCL	X'70'	SENDCMD/RCVCMD processor losing its ACB				
		USFRNOCE	X'71'	Operator command invalid				
				The following byte values apply to the FDBK field (RPLFDB3 in the IFGRPL DSECT):				
		USFIACT	X'00'	APPL is active				
		ISFIINA	X'04'	APPL is inactive				
		USFINA	X'08'	APPL never accepts logon requests				
		USFITNA	X'0C'	APPL temporarily not accepting logon Requests				
		USFIQUIE	X'10'	APPL no longer accepts logon requests				

Figure H-10 (Part 4 of 4). The RPL's RTNCD-FDBK-FDBK2 DSECT (ISTUSFBC)

Displacement **Control Block: NIB**

	Dec	*Hex*	Byte 0	Byte 1	Byte 2	Byte 3
	0	0				NIBLEN (NIBLEN)
	4	4	CID (NIBCID)			
	8	8	USERFLD (NIBUSER)			
	12	C	NAME (NIBSYM)			
	20	14	MODE (NIBMODE)			
See ISTDVCHR (Figure H-13)	28	1C	General characteristics	Device Type	Model	Additional characteristics
	32	20	Physical device address	DEVCHAR (NIBDEVCH)		
See ISTDPROC (Figure H-14)	36	24	PROC (NIBPROCD): PROC1	PROC2	PROC3	PROC4
	40	28	NIB attributes (NIBFLGS)		RESPLIM (NIBLIMIT)	
	44	2C	EXLST (NIBEXLST)			
	48	30	LOGMODE (NIBLMODE			
See ISTDBIND (Appendix J for DSECT at area printed to by BNDAREA)	56	38	BNDAREA (NIBNDAR)			
	60	3C	Reserved			

Figure H-11. The Format of the DOS/VS and OS/VS NIB

NIB DSECT: ISTDNIB

Field	*DSECT DS or ORG label*	*DSECT EQU label*	*Value*	*Meaning (For EQU, meaning when bit setting is on or when byte value is set)*	*DOS/VS Displacement Dec*	*DOS/VS Displacement Hex*	*OS/VS Displacement Dec*	*OS/VS Displacement Hex*
BNDAREA	NIBNDAR			Bind area address	58	38	56	38
CID	NIBCID	–	–	Communication ID	4	4	4	4
CON	NIBFLG1	NIBCON	X'40'	CON=YES	40	28	40	28
DEVCHAR	NIBDEVCH			(See ISTDVCHR, Figure H-13)	28	1C	28	1C
EXLST	NIBEXLST	–	–	EXLST address	44	2C	44	2C
LISTEND	NIBFLG1	NIBLAST	X'80'	LISTEND=NO	40	28	40	28
LOGMODE	NIBLMODE	–	–	LOGMODE value	48	30	48	30
MODE	NIBMODE	–	–	MODE value	20	14	20	14
NAME	NIBSYM	–	–	NAME value	12	C	12	C
NIBLEN	NIBLEN	–	–	NIB length	3	3	3	3
PROC	NIBPROCD			(See ISTDPROC, Figure H-14)	36	24	36	24
RESPLIM	NIBLIMIT	–	–	RESPLIM value	42	2A	42	2A
SDT	NIBFLG1	NIBSDAPP	X'20'	SDT=APPL	40	28	40	28
USERFLD	NIBUSER	–	–	USERFLD value	8	8	8	8

Figure H-12. The DOS/VS and OS/VS NIB DSECT (ISTDNIB)

DEVCHAR DSECT: ISTDVCHR

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
DEVCHAR	DEVSHCH			Device scheduling characteristics:	28	1C	28	1C
		DEVINPUT	X'80'	The device is an input device; it is capable of sending data to the application program.				
		DEVOTPUT	X'40'	The device is an output device; it is capable of receiving data sent to it from the application program.				
		DEVCONVR	X'20'	The device is equipped with the Conversational Mode feature. This means that the device can receive data instead of the usual positive response to the block of data just sent from the device.				
		DEVSUBND	X'10'	The device has schedulable Sub-Nodes				
		DEVSPS	X'08'	The device is a 3275 Display Station with a printer attached.				
		DEVNNSPT	X'04'	The device is connected to a dial-in or a dial-in-out line.				
		DEVCCTL	X'02'	Additional information is contained in the first half of the fourth byte (DEVFLAGS)				
		DEVRSV01	X'01'	Reserved				
	DEVTCODE			The device is the device listed or it is a device that is compatible with the one listed	29	1D	29	1D
		DEV2740	X'01'	a 2740 Communication Terminal.				
		DEV2741	X'02'	a 2741 Communication Terminal.				
		DEV1050	X'03'	a 1050 Data Communication System.				
		DEVTWX	X'04'	a CPT-TWX Teletypewriter Terminal.				
		DEVWTTY	X'05'	a World Trade Telegraph Station.				
		DEV115A	X'06'	a Western Union Plan 115 A Station				
		DEV83B3	X'07'	an AT&T 83B3 Selective Calling. Station.				
		DEV2715	X'08'	a 2715 Transmission Control Unit.				
		DEV2770	X'09'	a 2770 Data Communication Terminal.				
		DEV2780	X'0A'	a 2780 Data Transmisssion Terminal.				
		DEV3735	X'0B'	a 3735 Programmable Buffered Terminal.				
		DEV3780	X'0C'	a 3780 Data Transmission Terminal.				
		DEV1130	X'0D'	an 1130 CPU.				
		DEV1800	X'0E'	a 1800.				
		DEV3125	X'11'	a 370 CPU Model 125.				
		DEV3135	X'12'	a 370 CPU Model 135.				
		DEVSYS3	X'13'	a System/3 CPU or a System/32 CPU.				
		DEV2701	X'14'	a 2701 Data Adapter				
		DEV2703	X'15'	a 2703 Transmission Control Unit				
		DEV3704	X'16'	a 3704 Communications Controller.				
		DEV3705	X'17'	a 3705 Communications Controller.				
		DEV2980	X'18'	a 2980 General Banking Terminal.				
		DEV3277	X'19'	a 3277 Display Station.				
		DEV3284	X'1A'	a 3284 Printer.				
		DEV3286	X'1B'	a 3286 Printer.				
		DEV3275	X'1C'	a 3275 Display Station.				
		DEV3741	X'1D'	a 3741 Data Station (of a 3740).				
		DEV3747	X'1E'	a 3747 Data Converter (of a 3740).				

Figure H-13 (Part 1 of 3). The NIB's DEVCHAR DSECT (ISTDVCHR)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
		DEVMTA	X'28'	a Multiple Terminal Access (MTA) terminal. MTA terminals are explained in *Network Control Program Generation and Utilities.*				
		DEV2972	X'33'	a 2972 Station Control Unit.				
		DEV3271	X'34'	a 3271 Control Unit.				
		DEVCC	X'35'	a Cluster Controller.				
		DEV3272	X'36'	a 3272 Control Unit.				
		DEV1052	X'64'	a 1052 Printer-keyboard.				
		DEV1053	X'65'	a 1053 Printer.				
		DEV1054	X'66'	a 1054 Paper Tape Reader.				
		DEV1055	X'67'	a 1055 Paper Tape Punch.				
		DEV1056	X'68'	a 1056 Card Reader.				
		DEV1057	X'69'	a 1057 Card Punch.				
		DEV1058	X'6A'	a 1058 Printing Card Punch.				
		DEV1092	X'6B'	a 1092 Programmed Keyboard.				
		DEV1093	X'6C'	a 1093 Programmed Keyboard.				
		DEVLU	X'6D'	a Logical Unit (use record-mode).				
		DEV545	X'78'	a 545 Output Punch (of a 2770).				
		DEV1017	X'79'	a 1017 Paper Tape Reader (requires a 2772).				
		DEV1018	X'7A'	a 1018 Paper Tape Punch (requires a 2772).				
		DEV2203	X'7B'	a 2203 Printer (of a 2770).				
		DEV2213	X'7C'	a 2213 Printer (of a 2770).				
		DEV2265	X'7D'	a 2265 Display Station.				
		DEV2502	X'7E'	a 2502 Card Reader.				
		DEV50	X'7F'	a 50 Magnetic Data Inscriber (of a 2770).				
		DEV1255	X'80'	a 1255 Magnetic Character Reader.				
		DEV5496	X'81'	a 5496 Data Recorder.				
	DEVMCODE			Device model code:	30	1E	30	1E
		DEVMOD1	X'00'	Device is designated as Model 1				
		DEVMOD2	X'01'	Device is designated as Model 2				
	DEVFLAGS				31	1F	31	1F
		DEVFCCTL	X'F0'	used if device requires connection control.				
		DEVCBSC	X'80'	The device uses BSC line control.				
		DEVCSSL	X'40'	The device uses start-stop line control, but has no Transmit Interrupt feature.				
		DEVCRVB	X'20'	The device has the Transmit Interrupt feature; this means that the application program can interrupt a transmission from the device by issuing a RESET macro instruction.				
		DEVCSWL	X'10'	The terminal is connected via a switched line, rather than a leased line.				
		DEVCHAR3	X'0F'	Compatibility existing code.				
		DEVCATTN	X'08'	The terminal can interrupt the application program (and cause its ATTN exit-routine to be scheduled).				

Figure H-13 (Part 2 of 3). The NIB's DEVCHAR DSECT (ISTDVCHR)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
		DEVCCHEK	X'04'	The terminal has the Checking feature.				
		DEVCSTCL	X'02'	The terminal has the Station Control feature.				
		DEVCSLPN	X'01'	The terminal has the Selector Pen feature.				
DEVPHYSA		–	–	Physical device address e.g., 3270 "from" terminal for COPY operation.	32	20	32	20

Figure H-13 (Part 3 of 3). The NIB's DEVCHAR DSECT (ISTDVCHR)

PROC DSECT: ISTDPROC

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
PROC	PROPROC1	PROTRUNC	X'40'	PROC=TRUNC	36	24	36	24
		PROXPOPT	X'20'	=BINARY				
		PRODFASY	X'10'	=DFASYX				
		PRORESPX	X'08'	=RESPX				
		PROCA	X'04'	=CA				
		PROCS	X'02'	=CS				
		PRORPLC	X'01'	= RPLC				
	PROPROC2	PROERPO	X'40'	PROC=NERPOUT	37	25	37	25
		PROLGOT	X'20'	=NLGOUT				
		PRONTFL	X'04'	=NTMFLL				
		PROEMLC	X'02'	=ELC				
		PROCFTX	X'01'	=CONFTXT				
	PROPROC3	PROERPI	X'40'	PROC=NERPIN	38	26	38	26
		PROLGIN	X'20'	=NLGIN				
		PRONTO	X'10'	=NTIMEOUT				
		PROMONIT	X'04'	=MONITOR				
	PROPROC4	PROEIB	X'80'	PROC=EIB	39	27	39	27
		PROMODB	X'08'	=BLOCK				
		PROMODM	X'04'	=MSG				
		PROMODT	X'02'	=TRANS				
		PROMODC	X'01'	=CONT				

Figure H-14. The NIB's PROC DSECT (ISTDPROC)

APPENDIX I. START-STOP, BSC, AND LOCAL 3270 CONSIDERATIONS

This appendix lists various device-dependent aspects of programming with VTAM. There is a separate section for each supported type of terminal. Much of this information can be found elsewhere in this publication; it is repeated here for your convenience.

The devices listed in this appendix represent the level of support provided by VTAM for local non-SNA 3270, start-stop, BSC terminals. For any device that is not listed but that is compatible with a device that is listed, use the information given for the listed device.

See the appropriate terminal product component description or programming publication for VTAM considerations that apply to a particular SNA terminal product.

IBM 1050 Data Communication System

Before using the information in this section, you should be familiar with the description of this system in *IBM 1050 Reference Digest,* GA24-3020.

If FEATURE=ATTN is specified on the TERMINAL macro instruction during VTAM definition, the MONITOR processing option should be set in the NIB used for connection so that an ATTN exit list routine can be used to handle the attention interruptions.

If FEATURE=TOSUPPR is specified on the TERMINAL macro, the NTIMEOUT processing option should be set in the NIB used for connection.

INQUIRE (OPTCD=DEVCHAR) can be issued to determine the device type in the 1050 system only if the DEVICE operand is coded for the TERMINAL macro instruction during VTAM definition.

The KEEP processing option should be set if the application program's input area is too small to hold the data arriving from the communications controller. (The amount of incoming data is affected by the TRANSFER operand of the LINE macro and the CUTOFF operand of the GROUP macro.

Translation from lowercase to uppercase is not provided.

Idle characters are inserted by the communications controller (as specified with the LINESIZ and CRRATE operands of the LINE macro instruction). If the Auto-fill character generation feature is present, specify PROC=NTMFLL in the NIB used for connection; this will suppress the insertion of idle characters by the communications controller.

If you are using the 1050 in group mode, you will have one symbolic terminal name for issuing OPNDST and I/O requests. If you are using the 1050 in specific mode, Figure I-1 describes the handling for different types of lines under DOS/VS, OS/VS1, and OS/VS2 SVS.

When a 1050 dial-in device exceeds the negative response to polling limit, the NCP terminates the input operation and sets the error lock (first bit in FDBK2 set on). The error lock can be reset and the input operation retried. If the error persists, CLSDST should be issued.

Line / System	Multi-Point	Point-to-Point	Switched
DOS/VS	You define the 1050 and each of its components with unique symbolic names. To connect with a component, issue an OPNDST for it. The system handles scheduling in connections to more than one component.	You define the 1050 and each of its components with unique symbolic names. To connect with a component, issue an OPNDST for the terminal first and then the component. You handle scheduling in connections to more than one component by issuing an OPNDST for the terminal and then issuing the sequence OPNDST/requests/ CLSDST for one component at a time.	You can define only the 1050 with a unique symbolic name as a terminal. To connect with a a component, issue an OPNDST for the terminal and then supply it with the component selection character in the data stream. You handle scheduling in connections to more than one component by varying the selection character.
OS/VS	You define the 1050 and each of its components with unique symbolic names. To connect with a component, issue an OPNDST for it. The system handles scheduling in connections to more than one component.	You define the 1050 and each of its components with unique symbolic names. To connect with a component, issue an OPNDST for it. The system handles scheduling in connections to more than one component.	You can define only the 1050 with a unique symbolic name as a terminal. To connect with a component, issue an OPNDST for the terminal and then supply it with the component selection character in the data stream. You handle scheduling in connections to more than one component by varying the selection character.

Figure I-1. Handling Input/Output for 1050 Components Under DOS/VS and OS/VS

The MSG, LGIN, LGOUT, BINARY, and EIB processing options are invalid for 1050 devices.

After a READ macro instruction has been issued for an active 1050 terminal, it cannot be deactivated until data has been received.

IBM 2740 Communication Terminal, Model 1

Before using the information in this section, you should be familiar with the description of the 2740 in *IBM 2740 Communication Terminals Models 1 and 2 Component Description,* GA24-3403.

Translation from lowercase to uppercase is not provided.

If the terminal has no station control or transmit control feature, data can be entered by the terminal operator when the BID key is pressed, even though no READ or SOLICIT is pending for the terminal. The communications controller ignores the data. To avoid losing the data, verify that the installation has defined the terminal as conversational (CONV=YES on the TERMINAL macro) and obtain all data with WRITE (OPTCD=CONV) or READ (PROC=CONT) macro instructions.

If you are using the 2740 in group mode, you issue OPNDST and I/O requests for a single symbolic terminal name. If you are using the 2740 in specific mode, you issue OPNDST and I/O requests for symbolic terminal names representing *each* component of the system that you are addressing in specific mode. If the installation has provided you with the capability of using the 2740 in *either* mode, you should issue OPNDST for both the symbolic terminal name representing the entire 2740 system, *and* for the symbolic terminal name of the component. (If more than one component is on the same point-to-point line, you should establish connection with only one component at a time.)

The MSG, LGIN, LGOUT, EIB, BINARY, and MONITOR processing options are invalid for 2740-1 devices. BLOCK, NERPIN, and NERPOUT are valid only if the terminal has the checking feature.

IBM 2740 Communication Terminal, Model 2

Before using the information in this section, you should be familiar with the description of the 2740 in *IBM 2740 Communication Terminals Models 1 and 2 Component Description,* GA24-3403.

Regardless of the setting of the BLK-LBM-LBT option code when WRITE is issued, the output operation is done as though LBT had been set (that is, an EOT is sent to the terminal after the terminal receives the block of data and sends a positive response).

If a block of data sent to the terminal is too large to fit in the terminal's buffer, the application program is not notified of this fact.

If a terminal is addressed after a BID key has been pressed, leading graphic characters are returned to the application program indicating that the output operation could not be completed normally. If the LGOUT processing option is in effect, the leading graphic characters are placed in the WRITE RPL's SENSE field.

Translation from lowercase to uppercase is not provided.

If you are using the 2740 in group mode, issue OPNDST and I/O requests for a single symbolic terminal name. If you are using the 2740 in specific mode, issue OPNDST and I/O requests for the symbolic terminal names representing *each* component of the system that you are addressing in specific mode. If the installation has provided the capability of using the 2740 in *either* mode, you should issue OPNDST for both the symbolic terminal name representing the entire 2740 system, *and* for the symbolic terminal name of the component. (If more than one component is on the same point-to-point line, you should establish connection with only one component at a time.)

The MSG, NTMFLL, EIB, NTIMEOUT, MONITOR, ELC, and BINARY processing options are invalid for 2740-2 devices. BLOCK is valid only if the terminal has the checking feature.

IBM 2741 Communication Terminal

Before using the information in this section, you should be familiar with the description of the 2741 in *IBM 2741 Communication Terminal,* GA24-3415.

If FEATURE=ATTN is specified on the TERMINAL macro during VTAM definition, the MONITOR processing option should be set in the NIB used for connection so that an ATTN exit-routine can be used to handle the attention interruptions.

No text timeout limitation is provided for the terminal (that is, once EOA has been sent from the terminal, no time limit between successive data characters exists). The NTIMEOUT processing option should be specified.

SOLICIT (PROC=CONT) causes an EOA and an EOT to be sent to the terminal, placing it in a transmit state. Before issuing WRITE while soliciting data with CONT set, RESET (OPTCD=COND) must be issued.

Conversational output (WRITE with OPTCD=CONV) is valid for 2741 terminals, even though the terminal cannot be specified during VTAM definition as conversational.

If the terminal has no break feature, the first I/O request following connection should be READ or SOLICIT. The READ or SOLICIT may be completed in error if the terminal was powered on before the communications controller became active; if this occurs, issue RESET followed by WRITE to maintain the conversational mode of the 2741 terminal.

The LGIN, LGOUT, EIB, NERPIN, NERPOUT, and BINARY processing options are invalid for a 2741 terminal.

IBM Communicating Magnetic Card Selectric Typewriter

If FEATURE=ATTN is specified on the TERMINAL macro during VTAM definition, the MONITOR processing option should be set so that an ATTN exit-routine can be used to handle the attention interruptions.

If the terminal has no break feature, the first I/O request following connection should be READ or SOLICIT. The READ or SOLICIT may be completed in error if the terminal was powered on before the communications controller became active; if this occurs, issue RESET followed by WRITE to maintain the conversational mode of the terminal.

Text timeouts can be suppressed with the NTIMOUT processing option.

The LGIN, LGOUT, EIB, and BINARY processing options are invalid for this terminal.

IBM World Trade Telegraph Station (WTTY)

If FEATURE=ATTN is specified on the TERMINAL macro during VTAM definition, an ATTN exit-routine can be used to handle the attention interruptions. The MONITOR processing option must be set in order for the ATTN exit-routine to be scheduled.

If the MSG processing option is used for solicitation, the end-of-block sequence must be defined by the installation (with the GROUP macro). If TRANS is used, the end-of-transmission sequence must be defined by the installation (GROUP macro). CONT (continuous solicitation) can be used, but should be avoided if output for the terminal occurs frequently. If CONT is used, RESET must be issued before WRITE can be issued.

No terminal ID verification is provided.

Conversational writing (WRITE with OPTCD=COND) is valid for this terminal.

Text timeouts can be suppressed by setting the NTIMEOUT processing option for the terminal.

The LGIN, LGOUT, NTMFLL, EIB, NERPIN, NERPOUT, and BINARY processing options are invalid for this terminal.

IBM System/7 CPU

Before using the information in this section, you should be familiar with the information about System/7 in *MSP/7 Macro Library/Relocatable: Coding the Input/Output Macros,* GC34-0020.

The System/7 is supported as a 2740 (Model 1 with checking) on a switched or nonswitched point-to-point start-stop line or as a System/3 on a BSC line.

The BLOCK, MSG, LGIN, LGOUT, NTMFLL, EIB, NERPIN, NERPOUT, MONITOR, and BINARY processing options are invalid for the System/7.

A remote IPL of the System/7 is implemented by issuing a series of WRITE macro instructions to send the IPL object program text to the device. The ELC-NELC processing option must be set to ELC, and idle characters must not be used.

IPL on a Start-Stop Line

On a start-stop line, format the output data like this:

	E	E	U	D	L		E
First WRITE:	O	O	C	E	C	object program text	O
	T	A		L			B
					E		E
Successive WRITEs:					O	object program text	O
					A		B
					E		E
Last WRITE:					O	object program text	O
					A		T

(The EOT in the first WRITE is used to place the System/7 in stand-by mode in case it is not prepared to receive data.)

IPL on a BSC Line

On a BSC line, format the output data like this:

	D	D	E				
First WRITE:	C	C	N				
	1	1	Q				
	A						
System/7 Response:	C						
	K						
	0						
	D	S		D	E		E
Second WRITE:	L	T	$UBIPL code	L	T	or	T
	S	X		E	X		B
	A						
System/7 Response:	C						
	K						
	1						
	D	S		D	E		E
Successive WRITEs:	L	T	object program text	L	T	or	T
	E	X		E	X		B

System/7 Response:	ACK0	or	ACK1
Last WRITE:	EOT		

AT&T 83B3 Selective Calling Station

Conversational output (WRITE with OPTCD=CONV) is valid for this terminal.

Solicitation with BLOCK or MSG is handled by VTAM as though TRANS had been specified. If the CONT processing option is used, all WRITE requests must be preceded by RESET. Avoid CONT if WRITE requests are to be issued frequently.

If you are using the terminal in group mode, issue OPNDST and I/O requests to the symbolic terminal name representing the terminal. If you are using the terminalin specific mode, issue OPNDST and I/O requests to the various symbolic terminal names assigned to *each* component of the terminal. If the installation has provided the capability for using the terminal in *either* mode, you should issue OPNDST for the symbolic terminal name that represents the terminal, *and* for the symbolic terminal name that represents the terminal component. (If more than one component is on the same point-to-point line, you should establish connection with only one component at a time.)

The LGIN, LGOUT, NTMFLL, EIB, NTIMEOUT, NERPIN, NERPOUT, MONITOR, and BINARY processing options are invalid for this device.

CPT-TWX, Models 33 and 35 (TWX)

If FEATURE=ATTN is specified on the TERMINAL macro during VTAM definition, an ATTN exit-routine can be used to handle the attention interruptions. The MONITOR processing option must be set if the ATTN exit routine is to be scheduled.

Text timeouts can be suppressed by setting the NTIMEOUT processing option.

Solicitation with BLOCK or MSG is handled as though TRANS had been specified. If the CONT processing option is used, all WRITE requests must be preceded by RESET. Avoid CONT if WRITE requests are to be issued frequently.

Conversational output (WRITE with OPTCD=CONV) is valid for this terminal.

The LGIN, LGOUT, EIB, NERPIN, NERPOUT, and BINARY processing options are invalid for this device.

Western Union Plan 115A Station

Solicitation with BLOCK or MSG is handled by VTAM as though TRANS had been specified. If the CONT processing option is used, all WRITE requests must be preceded by RESET. Avoid CONT if WRITEs are to be issued frequently.

Conversational output (WRITE with OPTCD=CONV) is valid for this terminal.

If you are using the terminal in group mode, issue OPNDST and I/O requests to the symbolic terminal name representing the terminal. If you are using the terminal in specific mode, issue OPNDST and I/O requests for the symbolic terminal names representing each terminal component with which you wish to communicate. If the installation has provided the capability of using the terminal in either mode, you should issue OPNDST for both the symbolic terminal name that represents the terminal, and for the symbolic terminal name that represents the terminal component. (If more than one component is on the same point-to-point line, you should establish connection with only one component at a time.)

The LGIN, LGOUT, NTMFLL. EIB, NTIMEOUT, NERPIN, NERPOUT, MONITOR, and BINARY processing options are invalid for this device.

IBM 2770 Data Communication System

Before using the information in this section, you should be familiar with the description of the 2770 in *System Components: IBM 2770 Data Communication System,* GA27-3013.

By setting the ERASE-EAU-NERASE option code to ERASE, a WRITE macro instruction causes a ERASE/WRITE command to be sent to the system's 2265 terminal. ERASELBM or ERASELBT LDOs can also be used.

To send a WRITE at Line Address command to a 2265 terminal, include the WLA escape sequence in the data stream that is to be written to the terminal.

Error recovery messages from the 2770 system in the form of

S			S		E
O	%	S	T	text	T
H			X		X

are not recognized as such by VTAM or the communications controller, but are handled by VTAM as a text message with header. The communications controller removes the imbedded STX and passes the remainder to the application program as two blocks:

First block: % S Second block: text

When the application program receives the first block, bit 7 (DATAFLG=SOH) is set on, indicating header data. Another READ is required to receive the second block.

Test Request Messages (which begin SOH % /) are intercepted by the communications controller. The READ macro instruction that would have moved the message into program storage had the interception not occurred is canceled with RTNCD=12 and FDBK2=2.

Request for Test messages of the form

S				S		E
O	%	XYN	addr	T	text	T
H				X		X

are not intercepted, but are passed on to the application program in the same manner that any text message with header data would be passed: the application program's READ obtains the header portion, with the SOH removed and with bit 7 (DATAFLG=SOH) set on in the RPL's FDBK field; a second READ is required for the text portion.

If you are using the 2770 in group mode, you will have one symbolic name for issuing OPNDST and I/O requests. If you are using the 2770 in specific mode, input/output is handled as it is described for the 1050 Data Communications System in this appendix.

Data sent *from* the device in transparent text mode is placed in the application program's input area unaltered. The application program can check the NCP return codes in the extended response byte of the SENSE field to determine whether the data was sent in transparent text mode.

VTAM does not compress output data or expand input data; this must be done by the application program.

The LGIN, LGOUT, NTMFLL, NTIMEOUT, and MONITOR processing options are invalid for the 2770.

IBM 2780 Data Transmission Terminal

Before using the information in this section, you should be familiar with the description of the 2780 in *Component Description: IBM 2780 Data Transmission Terminal,* GA27-3005.

Device-control characters required for Vertical Forms Control must be supplied by the application program. The application program must likewise supply the control characters for Printer Horizontal Format Control (including the escape sequence ESC HT ...).

Test Request Messages (which begin SOH % /) are intercepted by the communications controller. The READ macro instruciton that would have moved the message into program storage had the interception not occurred is canceled with RTNCD=12 and FDBK2=2.

Request for Test messages of the form

```
S                    S          E
O   %   XYN   addr   T   text   T
H                    X          X
```

are not intercepted, but are passed on to the application program in the same manner that any text message with header data would be passed: the application program's READ obtains the header portion, with the SOH removed and with bit 7 (DATAFLG=SOH) set on in the RPL's FDBK field; a second READ is required for the text portion.

If you are using the 2780 in group mode, you will have one symbolic name for issuing OPNDST and I/O requests. If you are using the 2780 in specific mode, input/output is handled as it is described for the 1050 Data Communications System in this appendix.

Data sent *from* the device in transparent text mode is placed in the application program's input area unaltered. The application program can check the NCP return codes in the extended response byte of the SENSE field to determine whether the data was sent in transparent text mode.

The LGIN, LGOUT, NTMFLL, NTIMEOUT, and MONITOR processing options are invalid for 2780 devices.

IBM 2972 General Banking Terminal System, Models 8 and 11

Before using the information in this section, you should be familiar with the description of the 2972 system in *Component Description: IBM 2972 Models 8 and 11 General Banking Terminal Systems,* GL27-3020.

Request for Test messages of the form

S				S		E
O	%	XYN	addr	T	text	T
H				X		X

are not intercepted by the communications controller. Instead, VTAM passes the message to the application program in the same manner that any text message with header data is passed: the application program's READ obtains the header portion, with the SOH removed and with bit 7 (DATAFLG=SOH) set on in the RPL's FDBK field; a second READ is required for the text portion.

The Batched Message feature is not supported by VTAM.

VTAM removes the 1-byte station address from the input data before moving the data into the application program's input area.

If a 2980 (Model 1 or 4) Teller Station includes a passbook printer or a 2980 (Model 2) Administrative Station includes an auditor key, VTAM assigns a symbolic name to the passbook printer or auditor key. VTAM forms the name by prefixing a dollar sign ($) to the name of the 2980's TERMINAL entry and deleting the last character of the name. When the application program issues INQUIRE (OPTCD=TERMS) using the 2980's TERMINAL entry name, VTAM places the passbook printer or auditor key name in the NIB generated by INQUIRE.

The BLOCK, MSG, LGIN, LGOUT, NTMFLL, NTIMEOUT, and MONITOR processing options are invalid for 2972 devices.

BSC and Local IBM Non-SNA 3270 Information Display System (Record Mode)

Before using the information in this section, you should be familiar with the description of the 3270 system in *IBM 3270 Information Display System Component Description,* GA27-2749, and *Introduction to Programming the 3270,* GC27-6999.

The application program can communicate with a non-SNA 3270 as a remotely (BSC) or locally attached non-SNA terminal or as though it were a logical unit. The application program can be independent of the mode of attachment (local vs. remote) of a record-mode 3270. The terminal is treated as a logical unit by setting the NIB's MODE field to RECORD when the terminal is connected, and by exchanging messages and responses with SEND and RECEIVE macro instructions. None of the basic-mode macro instructions can be used.

Different devices on the same control unit may be used in different modes at the same time. A 3270 can be disconnected in one mode and reconnected in another.

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, restrictions apply to all SEND, RECEIVE, and SESSIONC communication with the 3270 terminal. These restrictions are described below. All aspects of communication (and connection) not mentioned apply as though the 3270 were a logical unit.

All commands and orders for the 3270 must be placed in the output data by the application program in the format expected by a BSC 3270 (that is, everything that follows the BSC communication control character is used as though the 3270 were a BSC 3270). Only remote command codes may be used even if the 3270 is attached locally.

A SEND macro instruction may point to a data area containing a Read Modified command to be sent to the device. The data retrieved from the device is placed in the application program's storage area in this format:

AID CUR[1] CUR[2] device control characters, orders, text

where AID is the Attention Identification and CUR[1] and CUR[2] form the 2-byte cursor address. If the terminal operator causes a "short read" to occur at the terminal (by pressing the CLEAR key or a Program Access key, for example), the input data consists of the AID only.

For OS/VS only, note that a negative response with SSENSE1=PATH will be generated if there are not enough buffers available to VTAM to read the full buffer or to assemble all the input for a RECEIVE.

When a Copy command is placed in the data stream, the application program must include the physical device address of the "from" device. This address can be obtained by issuing INQUIRE (OPTCD=DEVCHAR). See Appendix H for the location of the physical device address in DEVCHAR. Note that a Copy operation is not valid for a locally attached non-SNA 3270 terminal and should not be used in an application program that is intended to be attachment-independent.

No responses may be sent to the 3270. All incoming messages indicate that no response of any type is expected (RESPOND=(NEX,NFME,NRRN)).

Messages (requests) sent to the 3270 should contain only data. No change-direction indicators or data flow control commands should be sent (that is, only CONTROL=DATA is allowed). If the application program attempts to send one of these indicators, the SEND is completed with RTNCD=20 and FDBK2=71. Bracket indicators are used as described below. Chaining indicators must always mark the message as the sole element of a chain–that is, CHAIN=ONLY (all incoming messages are so marked).

No SESSIONC commands can be received from the 3270 terminal. Only the Clear command can be sent to it. The effect of the Clear command is to reset both incoming and outgoing sequence numbers to 0, terminate any current bracket, and allow data traffic to continue.

A limited bracket protocol must be used so that the application program and the terminal operator can resolve attempts to send messages simultaneously. In this protocol, either the application program or the 3270 can begin the bracket. Only the application program can end the bracket; the 3270 cannot.

If, at the beginning of a session, the application program is the first to send the begin bracket indicator, the next message from the 3270 will indicate that the bracket is being continued. (If the application program's attempt to start the bracket arrives at the 3270 after the 3270 has sent a begin bracket indicator, the application program's message is rejected by the 3270 with a sense indication that a bracket race error has occurred.) If the application program does not start the bracket at the beginning of a session, the first input message from the 3270 will contain the begin bracket indicator. All subsequent messages from the 3270 will indicate that the bracket is being continued.

Once a bracket is started, the bracket continues until the application program sends a Clear command or a data message containing an end bracket indicator. In either case, the bracket is ended, and either the application program or the 3270 can again attempt to begin a bracket, as at the beginning of a session.

With this protocol, no brackets need be ended within the session; the entire interval between connection and disconnection can be treated as one bracket. A local non-SNA 3270 uses bracket termination rule 2 (unconditional termination); a BSC 3270 uses bracket termination rule 1 (conditional termination).

If both the application program and the 3270 attempt to begin a bracket at the same time, if the application program attempts to begin a new bracket without ending the current one, the response to the application program's SEND indicates Request Reject (SSENSEI=RR). If the application program sends an EB or NBB bracket indicator while not in a bracket, a state error (SSENSEI=STATE) is returned.

When the SEND data stream contains a Read command, the resulting input is received as a separate message, not as a response to the SEND operation. The application program should not begin or end a bracket when the data being sent contains a Read command. The application program must request a definite response type 1 for each message sent to the 3270 that begins or ends a bracket. Definite responses type 1 is requested by setting RESPOND=(NEX,FME,NRRN) in the SEND's RPL. Definite responses type 1 should also be requested when a message is sent to a printer. All other output can indicate that either exception responses only–RESPOND=(EX,FME,NRRN)–or definite responses–RESPOND=(NEX,FME,NRRN)–are expected. Definite responses type 2 are not used.

Status and sense information may be available in the RPL's USENSEI field when an exception message or response is received. The format of the information shown below is the same as the 2-byte combined sense and status (S/S) format returned from a BSC 3270 except that the 2 leftmost bits of each byte are set to 0 (this makes it easier to design the application program to be independent of the local-remote attachment mode of the terminal).

First Byte[1]	Second Byte[1]	Meaning
08	00	Device busy
04	00	Unit busy
02	00	Device end
01	00	Transmission check
00	20	Command rejected
00	10	Intervention required
00	08	Equipment check
00	04	Data check or bus-out check
00	02	Control check
00	01	Operation check

[1]Only the rightmost 6 bits of each byte are significant. These bits may be set in combination and should be set individually.

Note that unit check occurs in the status of locally attached devices for all of the above conditions.

The SSENSEI (system sense) field is set following the receipt of negative responses, but the SSENSMI (system sense modifier) field is always set to 0. The SSENSEI field can be set to indicate a path error (SSENSEI=PATH), a state error (SSENSEI=STATE), a request reject (SSENSEI=RR), or no error (SSENSEI=0). See the description of the SSENSEI field near the end of Appendix C. In the case of SSENSEI=0, the USENSEI field should be used to determine the cause of the exception condition.

Logon requests for the 3270 cannot originate from the 3270 terminal itself (unlike a logical unit) except via the network solicitor.

If a 3270 device is polled and the control unit is not powered-on, an X'0C 01' return code will be received (see Appendix C). This will cause the terminal to be disconnected. When the 3270 device and 3271 control unit are powered-on, the user must call up the network operator and request connection with the network solicitor via a VARY LOGON command.

If a 3270 device is polled while it is powered-off and the control unit is powered-on, a device end status will be returned when the device is powered-on. When this occurs, reissue the request.

Test Request Messages from locally attached terminals are intercepted by VTAM. Test Request Messages from remotely attached terminals are intercepted by the communications controller. The arrival of the Test Request Message causes a SEND or RECIEVE macro instruction to be completed with an error return code (RTNCD=12, FDBK2=2). A clear operation is performed (as though the application program had sent a Clear command), the user's LOSTERM exit-routine (if available) is scheduled with a parameter list code of 12, and all pending communications are canceled. The application program should disconnect the terminal. Connection is established in the same manner as it was initially—either by the ACQUIRE or by the ACCEPT form of OPNDST.

Note; *Terminal operators of the 3270 should be advised that, unless they enter a logoff prior to actually shutting off the device, under certain circumstances, unauthorized users could gain access to the application program that they were running.*

BSC and Non-SNA Local IBM 3270 Information Display Station (Basic Mode)

Before using the information in this section, you should be familiar with the description of the 3270 system in *IBM 3270 Information Display System Component Description,* GA27-2749, and *Introduction to Programming the 3270,* GC27-6999.

The application program can communicate with a 3270 as a BSC terminal or as though it were a logical unit. The terminal is handled remotely as a BSC terminal or locally by setting the NIB's MODE field to BASIC when the terminal is connected, and by exchanging data with READ and WRITE macro instructions. None of the record-mode macro instructions can then be used. Different devices on the same control unit may be used in different modes at the same time. A 3270 device can be disconnected and reconnected in the other mode.

When the terminal sends sense and status information in response to a READ, WRITE, or DO macro instruction, VTAM places the information in the RPL's SENSE field. VTAM also sets the RPL's RTNCD field to 4 and sets the FDBK2 field to 2 to signal that the SENSE field has been set. The SENSE field codes are described below.

If the SENSE field indicates that an error arose because operator intervention was required at the device, the failed operation may be retried after execution of a RESET macro instruction with OPTCD set to UNCOND or LOCK.

If a 3270 device is polled and the control unit is not powered-on, a X'0C 01' return code will be received (see Appendix C). This will cause the terminal to be disconnected. When the 3270 device and control unit are powered-on, the user must call up the network operator and request connection with the network solicitor via a VARY LOGON command.

If a 3270 device is polled while it is powered-off and the control unit is powered-on, a device end status will be returned when the device is powered-on. When this occurs, reissue the request.

To unlock the keyboard of a 3270 display station, issue a WRITE macro instruction with an unlock-keyboard control character included in the data stream.

Test Request Messages (which begin SOH % /) from locally attached terminals are intercepted by VTAM. Test Request Messages from remotely attached terminals are intercepted by the communications controller. Any READ macro that would have received the message had the interception not occurred is completed with RTNCD=12 and FDBK2=2 return codes. The terminal must then be disconnected. In addition, the user's LOSTERM exit-routine (if one is available) is scheduled with a parameter list code of 40.

The BLOCK, MSG, CONT, LGIN, LGOUT, NTMFLL, EIB, NTIMEOUT, ELC, and MONITOR processing options are invalid for 3270 devices. BINARY is invalid for locally attached devices.

The BLK-LBM-LBT option code (applicable for output) should be set to LBT; BLK is invalid, and LBM requires that you be aware of whether the device is locally or remotely attached (because no line control characters are sent regardless of the attachment mode–see Appendix B).

When data is sent to a BSC 3270, any message containing an X'FF' embedded in it will fail. The X'FF' will result in a path error.

Only remote command codes may be used even if the non-SNA 3270 is attached locally.

Input Considerations

Since VTAM deletes all communication control characters arriving from remotely attached devices, your input processing need not take into account whether the device is locally or remotely attached.

To avoid losing incoming data when the input area is too small, specify the KEEP processing option in the NIB used to connect the device. Then if the data is too long to fit, VTAM will fill the input area to capacity, set the second bit (DATAFLG=EOB) of the FDBK field off, and hold the remaining data for the next read request. See the KEEP-TRUNC processing in the NIB macro instruciton for further details.

A READBUF LDO can be used to send a Read Buffer command to a terminal. See the LDO macro instruction (CMD=READBUF) for an explanation of how this is accomplished. The data in the application program's input area upon completion of the DO macro instruction is arranged like this:

AID CUR^1 CUR^2 SF   ATTR   text

where AID is the Attention Identification and CUR^1 and CUR^2 form the 2-byte cursor address. The SF (Start Field) and ATTR (Attribute Byte) are present only if the device buffer is formatted.

Output Considerations

There are three different output operations available; they are selected by setting the ERASE-EAU-NERASE option code in the RPL of a WRITE macro instruction.

WRITE (OPTCD=ERASE) is a two-part operation; it first clears the device's entire buffer, and then it sends the output data that you provide via the RPL's AREA field. In the beginning of that data you must provide the Write Control Character (WCC) followed by the appropriate device-control characters, orders, and text. If you set the BLK-LBM-LBT option code to LBT, you need not include line-control characters; VTAM will include them for remotely attached devices, and omit them for locally attached devices.

WRITE (OPTCD=EAU) sends an Erase All Unprotected command to the device. Since no output data is involved with this form of WRITE, set the RPL's RECLEN field to 0.

WRITE (OPTCD=NERASE) sends a Write command to the device. You must prepare the output data in exactly the same manner as is specified above for OPTCD=ERASE: begin the output data with WCC, followed by the appropriate device-control characters and orders.

WRITE (OPTCD=CONV) is not available for a 3270 in basic mode. WRITE (OPTCD=ERASE) should be used as described above.

For a READ/WRITE sequence, the write operation is not suspended pending the completion of the solicitation of the device. Instead, the write operation is completed and the solicitation of the device continues. This is true in all cases except when the READ is the first I/O operation. In that case, the write operation is suspended pending the completion of the solicitation of the device to establish a session.

Copy Considerations

With the COPYLBM LDO, an application program can send a "copy" command to copy the contents of a remotely attached 3277 Display Station to any other display station or printer connected to the same control unit. The COPYLBT LDO works like COPYLBM, except that after the data has been copied, VTAM waits for the "to" device's response and sends an EOT when the response is detected.

Note: *Since this facility is available only for remotely attached devices, you may wish to simulate a copy operation with READ and WRITE macro instructions. Using READ and WRITE macros allows you to use the same program code to handle copy operations for both locally and remotely attached devices.*

Specific information about using these LDOs is given in the DO and LDO macro instruciton descriptions. Briefly, the procedure is this: The LDO is built and its ADDR field set to point to a 3-byte area containing (1) a copy control character, and (2) the second 2 bytes of the "from" device's CID (in that order). A COPYLBT LDO must be used if the Start Print bit (bit 4) in the copy control character is set on. The LDO's LEN field must be set to 3. The address of the LDO is placed in the RPL's AREA field and the CID of the "to" device is placed in the ARG field. The RPL's ACB field must indicate the same ACB that was indicated by the RPL used to connect the "from" and "to" devices.

Note: *The operation will not work if the "from" device's buffer is locked against a copy operation. An application program can lock a "from" device's buffer by placing an attribute of Protected/Alphameric (hexadecimal 60) in buffer location 0 and setting the byte at buffer location 1 to zeros.*

Sense Information

When a READ, WRITE, or DO macro instruction is completed, the SENSE field may contain 2 bytes of status and sense information. If the SENSE field is extracted with SHOWCB, the 2 bytes are right-adjusted in the fullword work area. The possible hexadecimal values of the 2 bytes are:

First Byte[1]	Second Byte[1]	Meaning
C8	40	Device busy
C4	40	Unit specify
C2	40	Device end. This condition is reported if a powered-off 3270 display is powered on and a SOLICIT or READ with OPTCD=SPEC is outstanding for the terminal. The SOLICIT or READ should be reissued.
C1	40	Transmission check
40	C8	Equipment check
40	C4	Data check or bus-out check
40	C2	Control check
40	C1	Operation check
40	60	Command rejected
40	50	Intervention required

[1] Only the rightmost 6 bits of each byte are significant. These bits may be set in combination and should be tested individually.

Note that unit check occurs in the status of locally attached devices for all of the above conditions.

Security Consideration

Note: *Terminal operators of the 3270 should be advised that, unless they enter a logoff prior to actually shutting off the device, under certain circumstances, unauthorized users could gain access to the application program that they were running.*

IBM 3735 Programmable Buffered Terminal

Before using the information in this section, you should be familiar with the description of the 3735 terminal, in *IBM 3735 Programmer's Guide,* GC30-3001.

Data to be sent to the 3735 should be formatted in this manner:

Form Description Program message:

Preliminary message:	NUL	F	NUL	
Data block:	Unpacked FDP data block			
Ending message:	NUL	E	NUL	

Selectric message:	NUL	M	NUL	text
Terminate Communication Mode:	NUL	T	NUL	

	N		N	
Inquiry:	U	I	U	text
	L		L	

	N		N	
Power down:	U	P	U	
	L		L	

	N		N	
CPU ID list:	U	L	U	ID list
	L		L	

Each entry in the ID list contains the CPU ID of all CPUs that may communicate with the 3735 over switched lines. The CPU ID is defined by the user in the CUID operand of the BUILD macro instruction.

Status messages reporting abort conditions are sent from the terminal as

S	N		N			E
T	U	S	U	S^1	S^2	T
X	L		L			X

but when the block is placed in the application program's input area, the communication control characters are deleted:

N		N		
U	S	U	S^1	S^2
L		L		

When sending Form Description Program (FDP) data blocks to the terminal in ASCII transmission code, the last 6 bytes of the block must each be changed from FF to 7F or else deleted entirely. To remove your dependency on the transmission code used, it is recommended that you always remove the last 6 bytes (the sector flags) from FDP data blocks. You can accomplish this by specifying RECLEN=470 instead of RECLEN=476 in the RPL used for WRITE.

A READ macro instruction must be the first I/O request issued for a 3735 terminal following connection. All other requests are rejected until a READ is issued.

The NTMFLL, NTIMEOUT, and MONITOR processing options are invalid for the 3735 terminal.

IBM 3740 Data Entry System

Before using the information in this section, you should be familiar with the description of the 3740 system in *IBM 3740 Data Entry System Summary*, GA21-9152 (the order numbers of the 3741 and 3742 reference manuals are GA21-9183 and GA21-9184, respectively.

When the 3740 system sends sense data in response to READ, WRITE, or DO macro instructions, VTAM places the sense data in the RPL's SENSE field.

Request for Test messages of the form

S				S		E
O	%	XYN	addr	T	text	T
H				X		X

are not intercepted by VTAM, but are passed to the application program in the same manner as any text message with header data: the application program's READ obtains the header portion, with the SOH removed and with bit 7

(DATAFLG=SOH) set on in the RPL's FDBK field; a second READ is required for the text portion.

Data may be sent to the terminal in transparent text mode by specifying PROC=BINARY for the NIB used to connect the terminal. Data sent from the terminal in transparent text mode is placed in the application program's input area unaltered. The application program can determine that the data was sent in transparent text mode by examining the NCP return code in the extended response byte of the SENSE field.

The LGIN, LGOUT, NTMFLL, NTIMEOUT, and MONITOR processing options are invalid for 3740 devices.

IBM 3780 Data Communication Terminal

Before using the information in this section, you should be familiar with the description of the 3780 in *Component Information for the IBM 3780 Data Communication Terminal*, GA27-3063.

Data may be sent to the 3780 in transparent text mode by specifying PROC=BINARY for the NIB used to connect the terminal. Before sending the transparent text blocks on a point-to-point line, the communications controller first obtains the component selection character and inserts it into a block which it sends in nontransparent text mode. The component selection character is supplied by the installation in the ADDR operand of the COMP macro instruction.

Data sent from the device in transparent text mode is placed in the application program's input area unaltered. The application program can determine that the data was sent in transparent text mode by examining the NCP return code in the extended response byte of the SENSE field.

The application program must supply the control sequences required for Horizontal Format Control and Vertical Forms Control.

VTAM does not compress output data or expand input data.

Test Request Messages (which begin SOH % /) are intercepted by the communications controller. The READ macro instruction that would have moved the message into program storage had the interception not occurred is canceled with RTNCD=12 and FDBK2=2. The terminal must then be disconnected.

Request for Test messages of the form

S				S		E
O	%	XYN	addr	T	text	T
H				X		X

are not intercepted, but are passed on to the application program in the same manner that any text message with header data is passed: the application program's READ obtains the header portion, with the SOH removed and with bit 7 (DATAFLG=SOH) set on in the RPL's FDBK field; a second READ is required for the text portion.

Error recovery messages of the form

S			S		E
O	%	S	T	text	T
H			X		X

are likewise passed on to the application program as two blocks—the first is the header portion with the SOH removed, the second is the text portion, with the STX and ETX removed.

The LGIN, LGOUT, NTMFLL, NTIMEOUT, and MONITOR processing options are invalid for 3780 devices.

IBM System/3 CPU

The System/3 is supported as a BSC station on switched lines, and on both point-to-point and multipoint nonswitched lines. The EBCDIC and ASCII transmission codes are supported.

Data blocks received from the System/3 that contain both header data and text are handled as two blocks: the application program's READ obtains the header portion, with the SOH removed and with bit 7 (DATAFLG=SOH) set on in the RPL's FDBK field; a second READ is required for the text portion.

When the System/3's Continuous Conversation function is active, the CPU engages in a continuous exchange of write operations and conversational replies. To communicate with the System/3 in this manner, you must:

1. Specify PROC=MSG for the NIB used to connect the application program to the System/3 CPU
2. Specify OPTCD=CONV for all WRITE macro instructions
3. Issue WRITE with RECLEN=0 if no data is ready to be sent to the System/3 CPU
4. Be prepared to receive a reply of zero length

The READ, WRITE, WRITELBM, WRITELBT, WRTHDR, WRTPRLG, and WRTNRLG LDOs can be used with the System/3. See the description of the LDO macro instruction.

If leading graphic characters are received as a response to a WRITE macro instruction, the FDBK field is set to indicate this (DATAFLG=LG). The next READ macro instruction directed at the device will obtain the leading graphic characters. If leading graphic characters are received in response to a conversational WRITE (OPTCD=CONV), the leading graphic characters are placed in the data area indicated by the AAREA field of the RPL.

The NTMFLL and MONITOR processing options cannot be used with the System/3.

IBM System/370 CPU

The System/370 is supported on switched or nonswitched point-to-point BSC lines. The EBCDIC and ASCII transmission codes are supported.

Data blocks received from the System/370 that contain both header data and text are handled as two blocks: the application program's READ obtains the header portion, with the SOH removed and with bit 7 (DATAFLG=SOH) set on in the RPL's FDBK field; a second read is required for the text portion.

The READ, WRITE, WRITELBM, WRITELBT, WRTHDR, WRTPRLG, and WRTNRLG LDOs can be used with the System/370. See the description of the LDO macro instruction.

If leading graphic characters are received as a response to a WRITE macro instruction, the FDBK field is set to indicate this (DATAFLG=LG). The next READ macro instruction directed at the device will obtain the leading graphic characters. If leading graphic characters are received in response to a conversational WRITE (OPTCD=CONV), the leading graphic characters are placed in the data area indicated by the AAREA field of the RPL.

The NTMFLL and MONITOR processing options cannot be used with the System/370.

APPENDIX J. SPECIFYING SESSION PARAMETERS

When a VTAM application program issues an OPNDST macro instruction to establish connection with a logical unit, an SNA Bind command containing session parameters is created and sent to the logical unit. The session parameters provide both the application program and the logical unit information about the session and the protocols to be used. Using this information, they can determine whether or not they can operate together in the way in which the session parameters indicate. The session parameters, therefore, provide a method for both participants in a session to be aware of how they are to communicate with each other. For a description of some of the session parameters and how to use them in a VTAM application program, see the *VTAM Macro Language Guide.*

Session Parameter Fields

The contents of the session parameter fields are described in this section. To understand the nature of some of the bit settings, it is necessary to understand the difference between a primary logical unit and a secondary logical unit in an SNA network. The primary logical unit exercises a certain amount of control over the secondary logical unit. The secondary logical unit either simply responds to the commands of the primary or it requests the primary to do those things that it cannot do for itself. In VTAM, the application program is always the primary logical unit, and all the logical units with which an application program can communicate are the secondary logical units. Communication can flow from the primary to the secondary logical unit or from the secondary to the primary logical unit.

Note: *This Appendix is intended to describe the fields applicable to session parameters and is* not *intended to be used to locate fields in an SNA Bind command. Since the SNA Bind command is created by VTAM and not the application program, the Bind command contains fields set by VTAM which have no significance to the application program.*

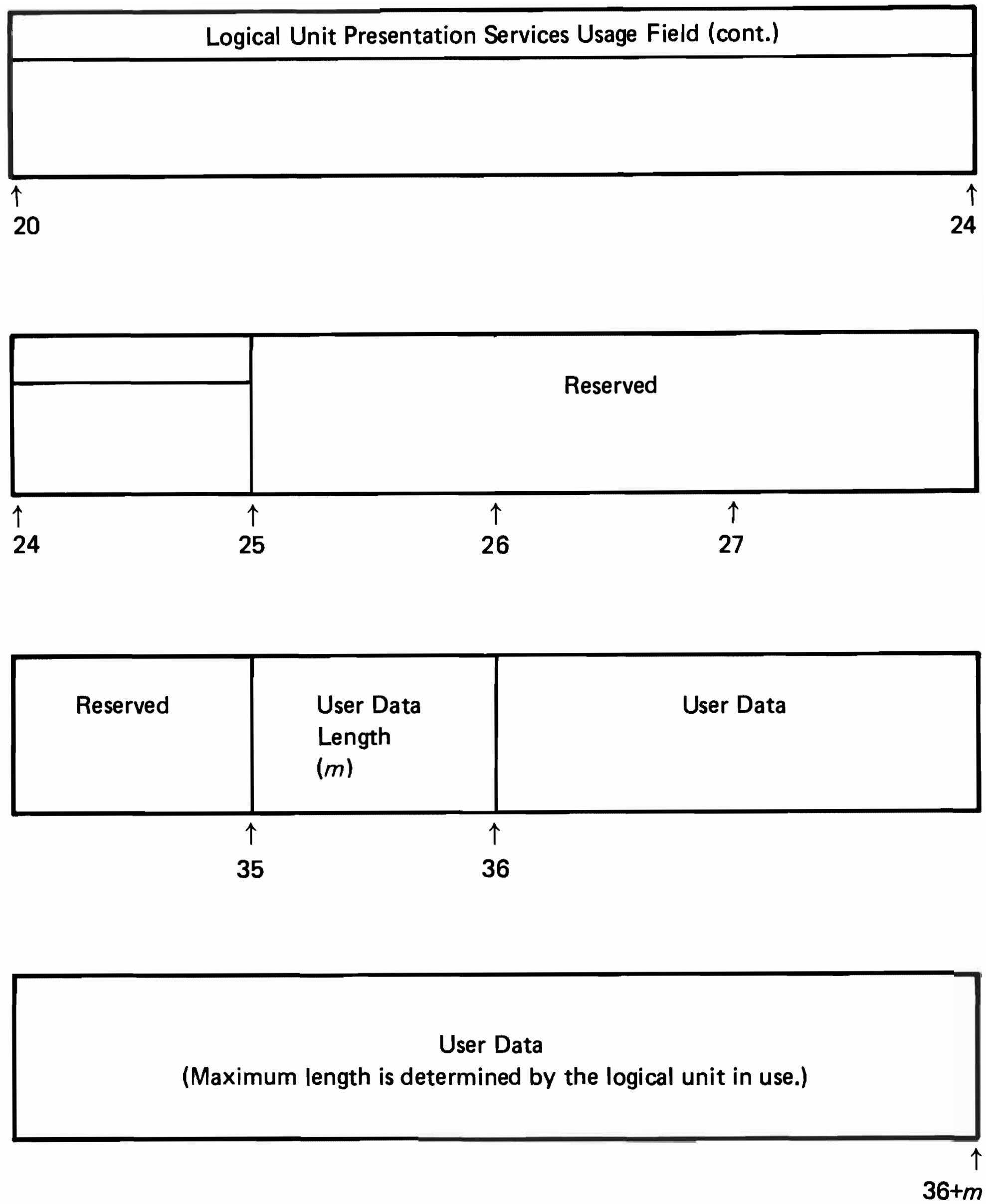
Bind Format	Reserved	Function Management Profile	Transmission Services Profile	FM Usage Field
				Primary Logical Unit Protocols
↑ 0		↑ 1	↑ 2	↑ 3 ... ↑ 4

Function Management Usage Field (cont.)		TS Usage Field
Secondary Logical Unit Protocols	Common Logical Unit Protocols	Reserved
↑ 4	↑ 5 ... ↑ 6	↑ 7 ... ↑ 8

Transmission Services Usage Field (cont.)			
Reserved	Secondary Logical Unit Request Unit Size	Primary Logical Unit Request Unit Size	Reserved
↑ 8	↑ 9	↑ 10	↑ 11 ... ↑ 12

TS Usage Field	Logical Unit Presentation Services Profile	LU Presentation Services Usage
Reserved		
↑ 12	↑ 13	↑ 14 ... ↑ 16

Logical Unit Presentation Services Usage Field (cont.)
(The format and content of this field are determined by the logical unit presentation services profile; see the description of this field for the formats that are available.)
↑ 16 ... ↑ 20



Function Management Profile

The function management profile identifies a predefined set of protocol rules. Each profile represents a different set of rules. Within each profile some of the protocol rules are mandatory, some are optional, and some do not apply. The profiles that have optional rules, require the use of the function management usage field to indicate how the optional rules are to be implemented. The profiles that are available to the VTAM application program and can be specified in the function management profile field are listed below.

Bit Setting (Byte 1) **0123 4567**	**Meaning**
0000 0010	Function management profile 2 is to be used. (This profile applies to 3270 terminals only.)
0000 0011	Function management profile 3 is to be used.
0000 0100	Function management profile 4 is to be used.

The protocol rules that apply to each of these function management profiles are:

Profile 2 (3270 Only)

The application program (primary logical unit):

- May not use multiple element chains
- May use either immediate or delayed request mode
- May request exception only, definite only, or both exception and definite responses to messages
- May not use compression
- May send End Bracket indicators if brackets are used
- May not use function management headers
- May use brackets
- Must use bracket termination rule 2 if brackets are used
- May use an alternate code
- Must use full duplex (FDX) transaction mode
- Must be responsible for error recovery
- May initiate brackets as bidder if brackets are used
- May not send Bid, Cancel, Chase, LUS, QC, QEC, RELQ, RSHUTD, RTR, SHUTC, SHUTD, or Signal data flow control commands
- May use immediate or delayed response mode

The logical unit (secondary logical unit):

- May not chain messages
- Must use delayed request mode
- Must not request responses to messages
- May not use compression
- Must not send End Bracket indicators if brackets are used
- May not use function management headers
- May use brackets
- Must use bracket termination rule 2 if brackets are used
- May use an alternate code
- Must use full duplex (FDX) transaction mode
- May not be responsible for error recovery
- May initiate brackets as first speaker if brackets are used
- May not send Bid, Cancel, Chase, LUS, QC, QEC, RELQ, RSHUTD, RTR, SHUTC, SHUTD, or Signal data flow control commands
- Must use immediate response mode

Profile 3

The application program (primary logical unit) and the logical unit (secondary logical unit):

- May use multiple element chains
- May use either immediate or delayed request mode

- May request exception only, definite only, both exception and definite, or no response to messages or chains of messages
- May use compression if function management headers are used
- May send End Bracket indicators if brackets are used
- May use function management headers
- May use brackets
- May use either bracket termination rule 1 or 2 if brackets are used
- May use an alternate code
- May use simplex, half duplex flip-flop, half duplex contention, or full duplex transaction mode
- May be responsible for error recovery (either the primary logical unit only or the sender may be responsible for error recovery)
- May initiate brackets if brackets are used (if the primary logical unit is to initiate brackets as first speaker, the secondary logical unit may only initiate brackets as bidder or if the secondary is first speaker, the primary must be bidder)
- May initiate change-direction protocol or win contention if a half duplex transaction mode is used (either the primary or the secondary logical unit, not both, may initiate change direction or win contention)
- May send Bid, Cancel, Chase, LUS, RSHUTD, RTR, SHUTC, SHUTD, or Signal data flow control commands, as appropriate
- May not send QC, QEC, or RELQ data flow control commands (that is, may not use quiesce protocol)
- Must use immediate response mode
- May use delayed control mode

Profile 4

The application program (primary logical unit) and the logical unit (secondary logical unit):

- May use multiple element chains
- May use either immediate or delayed request mode
- May request exception only, definite only, both exception and definite, or no response to messages or chains of messages.
- May use compression if function management headers are used
- May send End Bracket indicators if brackets are used
- May use function management headers
- May use brackets
- May use either bracket termination rule 1 or 2 if brackets are used
- May use alternate code
- May use simplex, half duplex flip-flop, half duplex contention, or full duplex transaction mode
- May be responsible for error recovery (either the primary logical unit or the sender may be responsible for error recovery)
- May initiate brackets if brackets are used (if the primary logical unit is to initiate brackets as first speaker, the secondary logical unit may only initiate brackets as bidder or if the secondary is first speaker, the primary must be bidder)

- May initiate change-direction protocol or win contention if a half duplex transaction mode is used (either the primary or secondary logical unit, not both, may initiate change direction or win contention)
- May send Bid, Cancel, Chase, LUS, QC, QEC, RELQ, RSHUTD, RTR, SHUTC, SHUTD, or Signal data flow control commands as appropriate
- Must use immediate response mode
- May use delayed control mode

Figure J-1 shows the bits in the function management usage field that are used to specify the protocol rules for a session. Figure J-2 summarizes the data flow control commands that can be used in each profile.

Logical Unit	Function Management Usage Field Bits	Profile 2[1]	Profile 3	Profile 4
Primary	Chaining Use	0	Opt	Opt
	Request Mode Selection	Opt	Opt	Opt
	Chain Response	Opt	Opt	Opt
	Compression Indicator	0	Opt	Opt
	Send End Bracket Indicator	1	Opt	Opt
Secondary	Chaining Use	0	Opt	Opt
	Request Mode Selection	1	Opt	Opt
	Chain Response	00	Opt	Opt
	Compression Indicator	0	Opt	Opt
	Send End Bracket Indicator	0	Opt	Opt
Common (Primary and Secondary)	FM Header Usage	0	Opt	Opt
	Brackets	Opt[2]	Opt	Opt
	Bracket Termination Rules	0	Opt	Opt
	Alternate Code Set Indicator	Opt	Opt	Opt
	FM Transaction Mode	00	Opt	Opt
	Recovery Responsibility	0	Opt	Opt
	Bracket First Speaker	0	Opt	Opt
	Contention Resolution	0	Opt	Opt

Key: Opt — This bit may be set to 0 or 1. The optional protocols may be further restricted by a logical unit profile that may also be set.

[1] This profile applies to 3270 terminals only.

[2] Brackets must be used for a local or BSC 3270 terminal in record mode.

Figure J-1. Bit Settings in the Function Management Usage Field for Each Function Management Profile

Logical Unit	Data Flow Control Command	Profile 2[1]	3	4
Common (Primary and Secondary)	Bid		Opt[2]	Opt[2]
	Cancel		Opt	Opt
	Chase		Opt	Opt
	LUS		Opt[3]	Opt[3]
	QC			Opt
	QEC			Opt
	RELQ			Opt
	RSHUTD		Opt[3]	Opt[3]
	RTR		Opt[2]	Opt[2]
	SHUTC		Opt[3]	Opt[3]
	SHUTD		Opt[4]	Opt[4]
	Signal		Opt	Opt

Key: Opt — This command may be used.
Blank— This command must not be used.

[1] This profile applies to 3270 terminals only.

[2] These commands may be used only if brackets are used.

[3] These commands are permitted from the secondary logical to the primary logical unit only.

[4] These commands are permitted from the primary to the secondary logical unit only.

Figure J-2. Data Flow Control Commands for Each Function Management Profile

Transmission Services Profile

The transmission services profile identifies a predefined set of session control commands that can be used in a session. Each profile represents a different set of session control commands. The profiles that are available to the VTAM application program and can be specified in the transmission services field are listed below.

Bit Setting (Byte 2) 0123 4567	**Meaning**
0000 0010	Transmission services profile 2 is to be used. (This profile applies to 3270 terminals only.)
0000 0011	Transmission services profile 3 is to be used.
0000 0100	Transmission services profile 4 is to be used.

The session control commands that apply to each of these profiles are shown in Figure J-3.

Session Control Command	Profile 2[1]	3	4
Clear	Req	Req	Req
RQR			Opt
SDT		Req	Req
STSN			Opt

Key: Req — This command must be used. The required commands may be issued by VTAM on behalf of the application program.

Opt — This command may be used.

Blank— This command must not be used.

[1] This profile applies to 3270 terminals qnly.

Figure J-3. Session Control Commands for Each Transmission Services Profile

Function Management Usage Field

The function management usage field is used to supplement the protocol rules defined by the function management profile. See the description of the function management profile field for an explanation of when and how to use this field. When the function management usage field is used, it is divided into three parts: the primary logical unit protocols (byte 3), the secondary logical unit protocols (byte 4), and the common logical unit protocols (bytes 5 and 6). The primary logical unit protocols apply only to the application program, the secondary logical unit protocols apply only to the logical unit, and the common logical unit protocols apply to both the application program and the logical unit in the same way. Bytes 3 and 4 have the same format; however, different bits may be on or off depending upon the protocol to be used and how it applies to the application program and the logical unit. The bits that can be set in the function management usage field are listed below.

Bit Setting (Bytes 3 and 4) 0123 4567	Meaning
0...	Chaining Use Bit. The primary or the secondary will send single element chains only.
1...	Chaining Use Bit. The primary or the secondary may send multiple element chains.
.0..	Request Mode Selection Bit. The primary or the secondary will use immediate request mode only (that is, only one request for a definite response will be outstanding at any one time); the sender will not send further normal-flow requests until the response has been received.

Bit Setting (Bytes 3 and 4) 0123 4567	Meaning
.1.. ...	Request Mode Selection Bit. The primary or the secondary will use delayed request mode (that is, one or more requests for definite responses may be outstanding at any one time). The responder may use immediate or delayed response mode (that is, the requests for definite responses may be responded to in any order depending on the function management profile that was specified).
..00	Chain Response Bits. The primary or the secondary does not require any responses to chains.
..01	Chain Response Bits. The primary or the secondary requires only exception responses to chains.
..10	Chain Response Bits. The primary or the secondary requires definite responses to chains.
..11	Chain Response Bits. The primary or the secondary may request either definite or exception responses to chains.
.... ..0.	Compression Indicator Bit. The primary or the secondary will not use compression (that is, remove extraneous blank characters from data before it is sent).
.... ..1.	Compression Indicator Bit. The primary or the secondary may use compression (that is, remove extraneous blank characters from data before it is sent). This bit setting may not be used if the function management header bit is set to 0.
....0	Send End Bracket Indicator Bit. The primary or the secondary will not send an End Bracket indicator. If the brackets bit is set to 1, this bit may not be set to 0 in both byte 3 and byte 4.
....1	Send End Bracket Indicator Bit. The primary or the secondary may send an End Bracket indicator. This bit setting may not be used if the brackets bit is set to 0.
.... xx..	Reserved.

Bit Setting (Byte 5) 0123 4567	Meaning
.0..	Function Management Header Bit. The primary and the secondary will not use function management headers.
.1..	Function Management Header Bit. The primary and the secondary may use function management headers.
..0.	Brackets Bit. The primary and the secondary will not use brackets.
..1.	Brackets Bit. The primary and the secondary may use brackets.
...0	Bracket Termination Rules Bit. The primary and the secondary will use bracket termination rule 2 (that is, the bracket will be unconditionally terminated when the chain that indicates the end of a bracket is processed). This bit must be set to 0 if the brackets bit is set to 0.

Bit Setting (Byte 5) 0123 4567	Meaning
...1	Bracket Termination Rules Bit. The primary and the secondary will use bracket termination rule 1 (that is, when terminating an existing bracket, the bracket will be terminated unconditionally unless the last message of a chain that indicates the end of a bracket requests a definite response. In that case the bracket will not be terminated unless a positive response is processed). This bit setting may not be used if the brackets bit is set to 0.
.... 0...	Alternate Code Set Bit. Only standard code will be used in this session.
.... 1...	Alternate Code Set Bit. Either the standard or the alternate code will be used in this session.
x... .xxx	Reserved.

Bit Setting (Byte 6) 0123 4567	Meaning
00..	Function Management Transaction Mode Bits. The primary and the secondary will use full duplex (FDX) communication.
01..	Function Management Transaction Mode Bits. The primary and the secondary will use half duplex contention (HDX-CON) communication. The contention resolution bit may also be set.
10..	Function Management Transaction Mode Bits. The primary and the secondary will use half duplex flip-flop (HDX-FF) communication. The contention resolution bit may also be set.
11..	Function Management Transaction Mode Bits. The primary and the secondary will use simplex (master/slave) communication.
..0.	Recovery Responsibility Bit. The primary is responsible for error recovery.
..1.	Recovery Responsibility Bit. The sender is responsible for error recovery.
...0	Brackets First Speaker Bit. The secondary will act as first speaker.
...1	Brackets First Speaker Bit. The primary will act as first speaker.
....0	Contention Resolution Bit. The secondary will send data first after an SDT (HDX-FF) by sending data or the secondary will win a contention with the primary (HDX-CON).
....1	Contention Resolution Bit. The primary will send data first after an SDT (HDX-FF) by sending data or the primary will win a contention with the secondary (HDX-CON).
.... xxx.	Reserved.

Transmission Services Usage Field

The transmissison services usage field specifies the maximum length of a chain element (this corresponds to a request unit, which is only data) that can be sent by the application program (primary logical unit) or the logical unit (secondary logical unit) with which the application program can communicate. This information is divided into two parts: the secondary logical unit send request unit size (byte 9) and the primary logical unit send request unit size (byte 10). Both bytes have the same format; however, they may be set to different values depending upon the requirements of the application program and the logical unit. The bits that can be set are listed below.

Bit Setting (Bytes 9 and 10) 0123 4567	Meaning
0000	There is no maximum length specified.
1000	This is the mantissa (m) of the formula $m(2^n)$ used to determine the maximum length of requests that can be sent by the primary or secondary.
.... 0000	This is the exponent (n) of the formula $m(2^n)$ used to determine the maximum length of request units that can be sent by the primary or secondary.

Logical Unit Presentation Services Profile

The logical unit presentation services profile identifies a predefined type of logical unit. Each profile represents a different type of logical unit that uses a unique subset of the SNA defined protocols and data streams for its operation. Both the primary and the secondary logical units must be able to operate with the special functions if they are to be used. The logical unit presentation services usage field is used in conjunction with the presentation services profile field. The profile field determines the format of the usage field; the usage field is used to specify optional protocols or data streams that are applicable to the profile. The logical unit presentation services profiles that can be specified are listed below.

Bit Setting (Byte 13) 0123 4567	Meaning
.000 0000	Logical unit presentation service profile 0 is used.
.000 0001	Logical unit presentation service profile 1 is to be used.
.000 0010	Logical unit presentation service profile 2 is to be used.
.000 0011	Logical unit presentation service profile 3 is to be used.
x...	Reserved.

The subsets of protocols and data streams that apply to each of these logical unit presentation services profiles are:

Profile 0

Use of a set of protocols and data streams agreed upon by the logical units involved. SNA does not define them.

Profile 1

- Use of predefined function management headers to identify a variety of data set types and controls that can be sent and received
- Use of a SNA character string (SCS) and a defined set of SNA protocols for a keyboard/printer device

Profile 2

- Use of a 3270 data stream and a defined set of SNA protocols for a keyboard/display device

Profile 3

- Use of a 3270 data stream and a defined set of SNA protocols for a printer

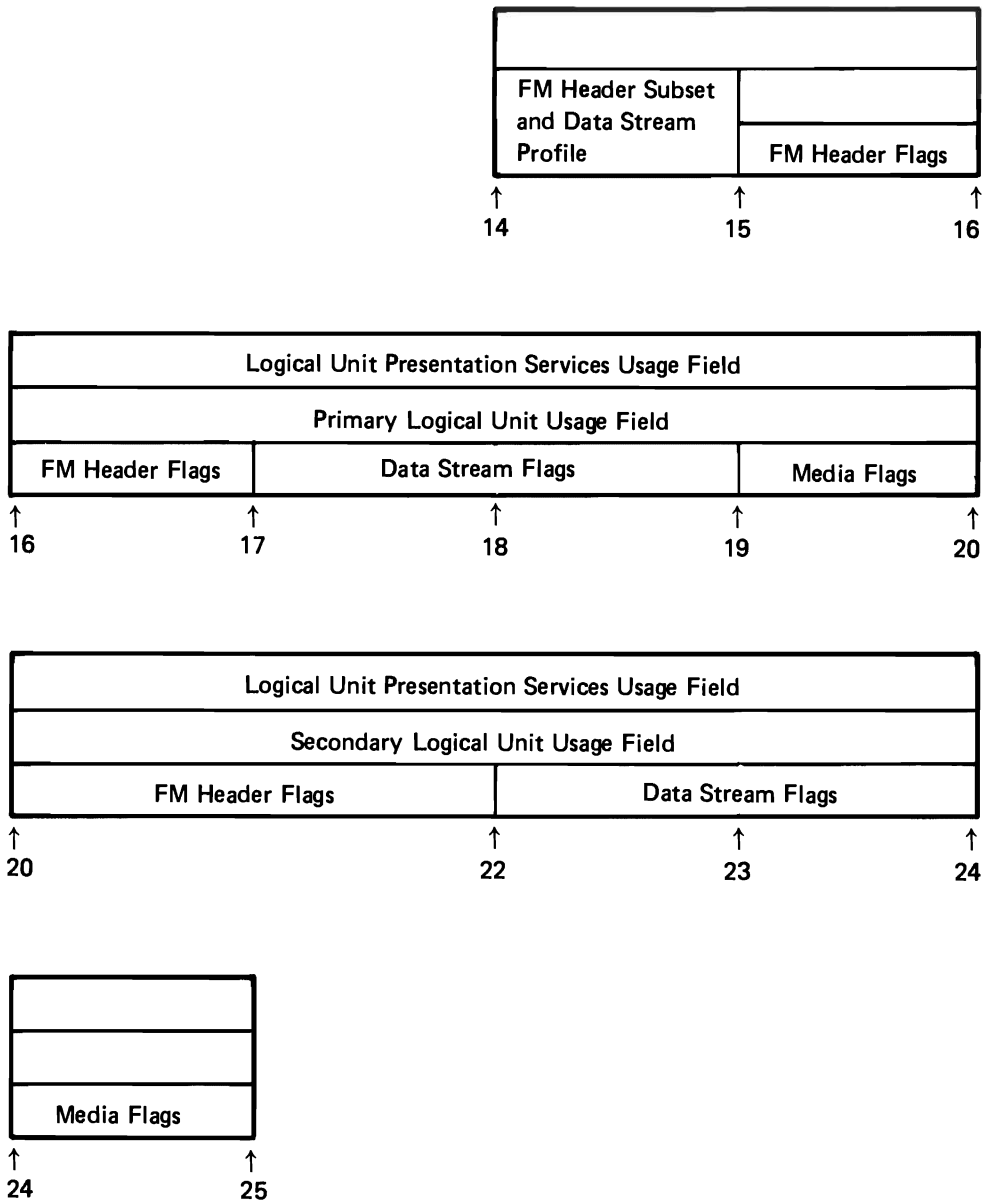
Logical Unit Presentation Services Usage Field

The logical unit presentation services usage field contains information describing the subset of SNA protocols and data streams for the type of logical unit being used. Each logical unit has a presentation services profile associated with it. This profile identifies any of the special functions that may be available and determines the format of this field. See the description of the logical unit presentation services profile field for an explanation of the profiles. The format of the logical unit presentation services usage field that corresponds to each profile is as follows:

Profile 0

Undefined

Profile 1



Function Management Header Subset and Data Stream Profile: This field identifies a predefined set of function management header and data stream subsets. Each subset specifies the types of data sets and data set controls and the types of data streams that can be handled. The subset profiles that are available are listed below.

Bit Setting (Byte 14) 0123 4567	Meaning
0001	Function management header subset 1 is to be used.
0010	Function management header subset 2 is to be used.
0011	Function management header subset 3 is to be used.
.... 0000	Data stream subset 0 is to be used.
.... 0001	Data stream subset 1 is to be used.

Function Management Header Subset Flags: These flags identify the types of data set and data sets controls that can be used. They are specified for the primary logical unit (bytes 15 and 16) and for the secondary logical unit (bytes 20 and 21). Both sets of bytes have the same format; however, different bits may be set for the primary and secondary. If the function management usage bit in the function management usage fields is set to 0, these flags may not be used and should be set to 0. The function management header subset flags that can be set are listed below.

Bit Setting (Bytes 15 or 20) 0123 4567	Meaning
Function Management Header Subset 1, 2, or 3	
0...	The transmission may be interrupted once to send to another logical unit (two destinations are outstanding).
1...	The transmission may be interrupted twice to send to another logical unit (three destinations are outstanding).
.0..	The primary and the secondary must not send compacted data.
.1..	The primary and the secondary may send compacted data. If this bit is used, it must be set to 1 for both the primary and the secondary.
..0.	The primary and the secondary must not send PDIR.
..1.	The primary and the secondary may send PDIR.
...x xxxx	Reserved (for subsets 1 and 2 only)
Function Management Header Subset 3	
...0	The primary or the secondary will not send keyed direct data sets with Add or Erase Record headers using RECID headers.
...1	The primary or the secondary may send keyed direct data sets with Add or Erase Record headers using RECID headers.
.... 0...	The primary or the secondary will not send sequential data sets with Add.
.... 1...	The primary or the secondary may send sequential data sets with Add.
.... .0..	The primary or the secondary will not send addressed direct data sets that are accessed sequentially with Add, Note, and Note Reply in the opposite direction.
.... .1..	The primary or the secondary may send addressed direct data sets that are accessed sequentially with Add, Note, and Note Reply in the opposite direction.
.... ..0.	The primary or the secondary do not support series IDs (with status in reply).
.... ..1.	The primary or the secondary support series IDs (with status in reply).
....0	The primary or the secondary does not support Add Replicate and Replace Replicate.
....1	The primary or the secondary supports Add Replicate and Replace Replicate.

Bit Setting (Bytes 16 or 21) 0123 4567	**Meaning**
Function Management Header Subset 1 or 2	
xxxx xxxx	Reserved.
Function Management Header Subset 3	
.0..	The primary or the secondary does not support query for data sets.
.1..	The primary or the secondary supports query for data sets.
..0.	The primary or the secondary does not support Create, Scratch, and Scratch All data sets.
..1.	The primary or the secondary supports Create, Scratch, and Scratch All data sets.
...0	The primary or the secondary does not support Execute FP.
...1	The primary or the secondary supports Execute FP.
x... xxxx	Reserved.

Data Stream Subset Flags: These flags identify the type of data stream that can be used. They are specified for the primary logical unit (bytes 17 and 18) and for the secondary logical unit (bytes 22 and 23). Both bytes have the same format; however, different bits may be set for the primary and secondary. The data stream subset flags that can be set are listed below.

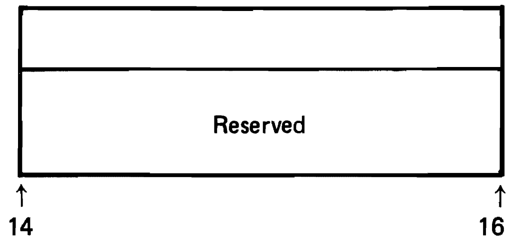
Bit Setting (Bytes 17 or 22) 0123 4567	**Meaning**
0...	The primary or the secondary will not send an interactive data stream (BS, CR, LF, HT, VT, ENP, and INP).
1...	The primary or the secondary may send an interactive data stream (BS, CR, LF, HT, VT, ENP, and INP).
.0..	The primary or the secondary will not send a horizontal format data stream (SHF).
.1..	The primary or the secondary may send a horizontal format data stream (SHF).
..0.	The primary or the secondary will not send a vertical tab data stream (SVF).
..1.	The primary or the secondary may send a vertical tab data stream (SVF).
...0	The primary or the secondary will not send a vertical select data stream (SVF(*channels*) and SEL).
...1	The primary or the secondary may send a vertical select data stream (SVF(*channels*) and SEL).
....0	The primary or the secondary will not send a transparency data stream (TRN and IRS).
....1	The primary or the secondary may send a transparency data stream (TRN and IRS).
.... xxx.	Reserved.

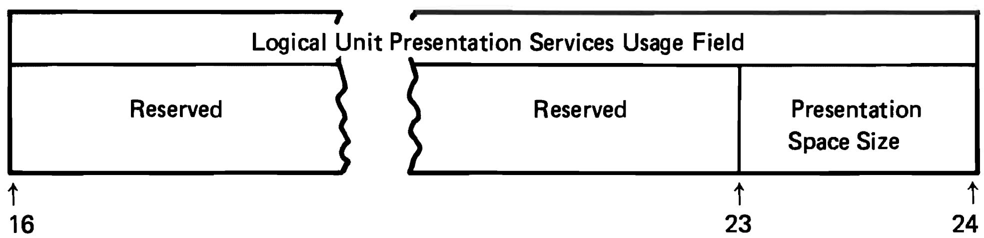
Bit Setting (Bytes 18 and 23) 0123 4567	Meaning
xxxx xxxx	Reserved.

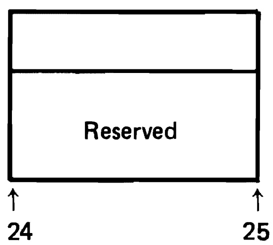
Media Flags: These flags identify the types of physical recording media for which the data stream can be formatted. They are specified for the primary logical unit (byte 19) and for the secondary logical unit (byte 24). Both bytes have the same format; however, different bits may be set for the primary and secondary. The media flags that can be set are listed below.

Bit Setting (Bytes 19 or 24) 0123 4567	Meaning
0...	The primary or the secondary will not use document output.
1...	The primary or the secondary may use document output.
.0..	The primary or the secondary will not use card format.
.1..	The primary or the secondary may use card format.
..0.	The primary or the secondary will not use exchange media format.
..1.	The primary or the secondary may use exchange media format.
...0	The primary or the secondary will not use disk, data management format.
...1	The primary or the secondary may use disk, data management format.
.... xxxx	Reserved.

Profile 2



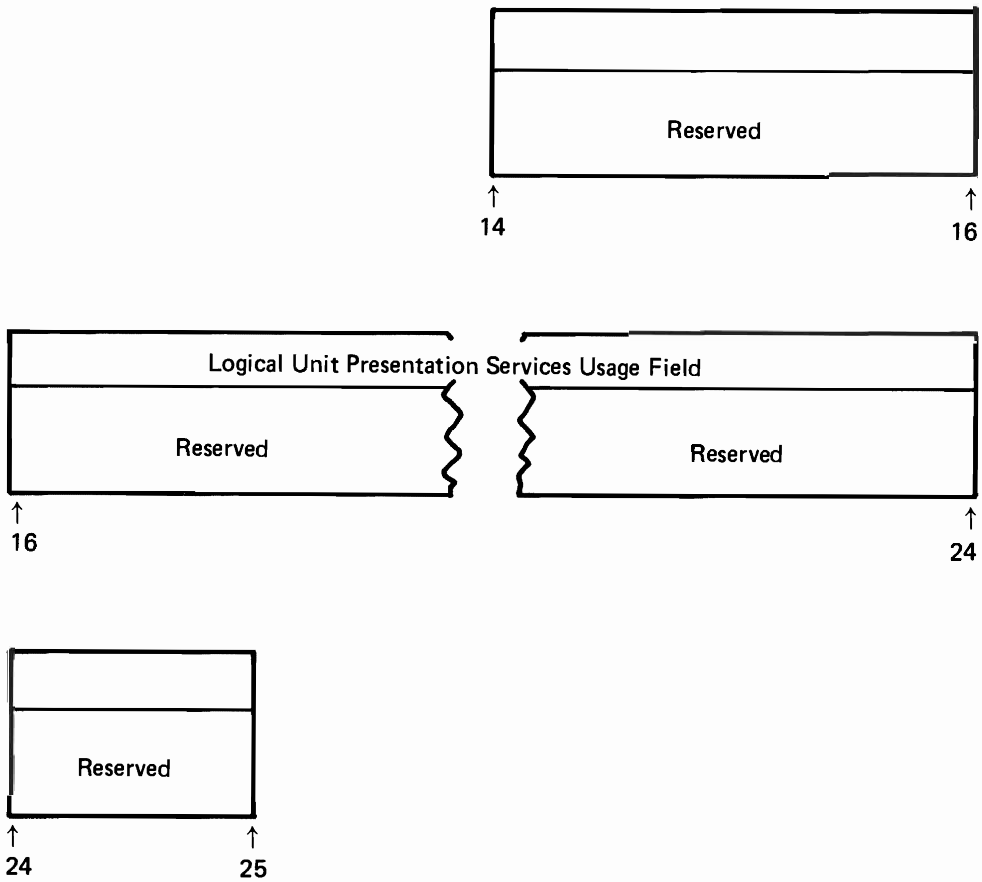




Presentation Space Size: The presentation space size field identifies the size of the screen matrix for display devices. The matrix sizes that can be specified are listed below.

Bit Setting (Byte 23) 0123 4567	Meaning
0000 0000	Matrix size is not defined.
0000 0001	A matrix size of 12 by 40 bytes is to be used.
0000 0010	A matrix size of 24 by 80 bytes is to be used.

Profile 3



Logical unit presentation services profile 3 does not require the use of this field.

User Data Length

This field specifies the length of the user data field. The value contained in this field can be any hexadecimal value from X'00' to X'FF'. If X'00' is specified there is no user data.

User Data

This field can be used to send data to the secondary logical unit as a part of the Bind command.

The Bind Area Format and DSECT

This section describes the format and DSECT (ISTDBIND) of the bind area that can be used in a VTAM application program to create the session parameters that will accompnay an OPNDST. Using the bind area, the application program can override or change any of the session parameters that may have been obtained from a logon mode table or a pending logon. This gives the application program the ability to tailor predefined session parameters to its own needs before initiating connection with the logical unit.

The bind area is identified by the BNDAREA field of the NIB. It may be either an area in the application program specifically created for this purpose or it may be the AREA field of the RPL that was used by an INQUIRE with OPTCD=SESSPARM macro instruction. The format map and DSECT description are an aid to examining or setting up the contents of a bind area (Figures J-4 and J-5). The IBM-supplied DSECT ISTBIND is provided as part of the system macro library (the source statement library in DOS/VS or SYS1.MACLIB in OS/VS). The DSECT can be included in a VTAM application program by including the ISTDBIND macro instruction in the program.

A DSECT is an overlay (map) containing labels that correspond to field displacements, bit settings, and byte values. A field displacement is the displacement of a field from the beginning of the bind area. It is defined by the DS (or ORG) instructions in the DSECT. A bit setting is an assembler EQU instruction (such as LABEL1 EQU X'80') that identifies a particular bit or set of bits. The label in the EQU instruction could be used as the immediate data in a TM instruction, for example. A byte value is also an assembler EQU instruction (such as LABEL2 EQU X'23') that identifies a particular value in a byte. This label could be used as the immediate data in a CLI instruction, for example.

The general manner in which DSECTs are used is described in "The DSECT Instruction" section of *OS/VS and DOS/VS Assembler Language*, GC33-4010.

To avoid the risk of duplicating DSECT labels in your program, avoid using any label that begins with the characters BIN. Be very careful to set all relevant bits and fields. You may notice that the bind area DSECT is more extensive than is shown here. The format maps and DSECT descriptions in this section do not include those fields that are set by VTAM and not by the application program or that are currently undefined in SNA; all such fields are marked "reserved".

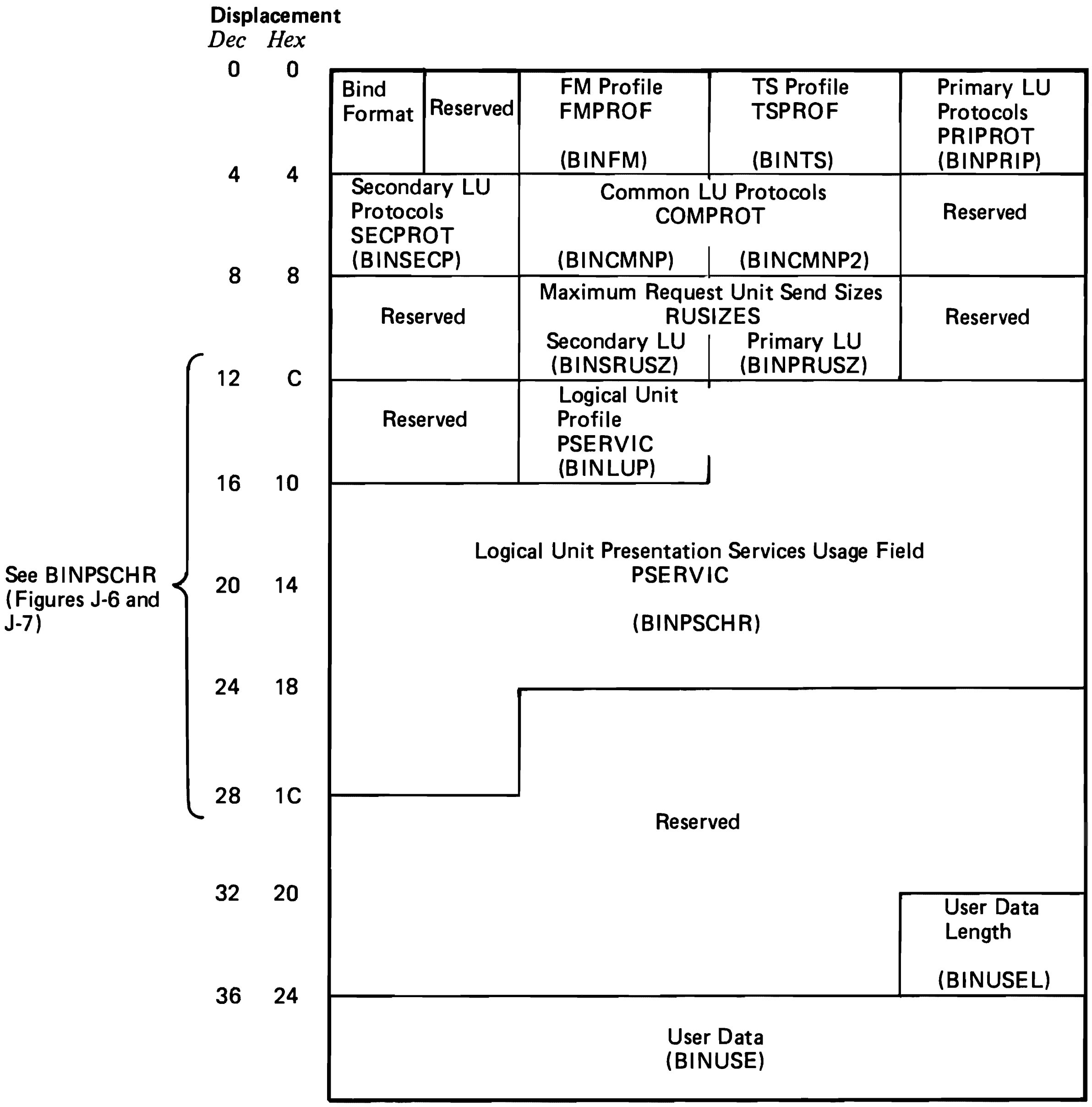


Figure J-4. The Format of the BNDAREA (ISTDBIND)

BNDAREA DSECT: ISTDBIND – FORMAT 0

Field	*DSECT DS or ORG label*	*DSECT EQU label*	*Value*	*Meaning (For EQU, meaning when bit setting is on or when byte value is set)*	*DOS/VS Displacement Dec*	*DOS/VS Displacement Hex*	*OS/VS Displacement Dec*	*OS/VS Displacement Hex*
Bind Format and Type	BINFMTY	BINFMT0	X'00'	Bind format (This half-byte is set by VTAM)	0	0	0	0
		BINTYPE		Bind type (This half-byte is reserved)				
FM Profile	BINFM				1	1	1	1
TS Profile	BINTS	BINTS0	X'00'	ID only and no reset state	2	2	2	2
		BINTS1	X'01'	ID only and no reset state				
		BINTS2	X'02'	Sequence numbers and no reset state				
		BINTS3	X'03'	Sequence numbers and reset state				
		BINTS4	X'04'	Sequence numbers and reset state (STSN and RQR)				
Primary LU Protocol	BINPRIP	BINPCHN	X'00'	Only single message chain allowed	3	3	3	3
			X'80'	Multiple message chain allowed				
		BINPCHNR						
		BINNORSP	X'00'	No response				
		BINEXRSP	X'10'	Exception responses				
		BINDFRSP	X'20'	Definite response				
		BINNYRSP	X'30'	Definite or exception response				
		BINPMCH	X'00'	Single outstanding chains only; immediate request mode				
			X'40'	Multiple outstanding chains; delayed request mode				
		BINPCMP	X'00'	Compression must not be used				
			X'02'	Compression may be used				
		BINPSEB	X'00'	Primary will not send End Bracket indicator				
			X'01'	Primary may send End Bracket indicator				
Secondary LU Protocol	BINSECP	BINSCHN	X'00'	Single message in chain only	4	4	4	4
			X'80'	Multiple message in chain				
		BINSCHNR						
		BINNORSP	X'00'	No response				
		BINEXRSP	X'10'	Exception responses				
		BINDFRSP	X'20	Definite response				
		BINNYRSP	X'30'	Definite or exception responses				
		BINSCMP	X'00'	Compression must not be used				
			X'02'	Compression may be used				
		BINSMCH	X'00'	Immediate request mode				
			X'40'	Delayed request mode				
		BINSSEB	X'00'	Secondary will not send End Bracket indicator				
			X'01'	Secondary may send End Bracket indicator				
Common LU Protocol	BINCMNP	BINALT	X'00'	Alternate code must not be used	5	5	5	5
			X'08'	Alternate code may be used				
		BINBKTR	X'00'	Unconditional bracket termination (Termination Rule 2)				
			X'10'	Conditional bracket termination (Termination Rule 1)				
		BINBRAK	X'00'	Brackets will not be used				
			X'20'	Brackets will be used				

Figure J-5 (Part 1 of 2). The BNDAREA DSECT (ISTDBIND)

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
		BINFMHD	X'00'	Function Management Headers must not be used				
			X'40'	Function Management Headers may be used				
	BINCMNP2	BINBKFS	X'00'	Secondary is first speaker in bracket mode	6	6	6	6
			X'10'	Primary is first speaker in bracket mode				
		BINCONR	X'00'	For half-duplex flip-flop mode, secondary is the first speaker when the data traffic reset is lrft; for half-duplex contention mode, secondary wins contention				
			X'01'	For half-duplex flip-flop mode, primary is the first speaker when the data traffic reset state is left; for half-duplex contention mode, primary wins contention				
		BINFMTRM						
		BINFLDPX	X'00'	Full-duplex transaction mode				
		BINHDXC	X'40'	Half-duplex contention mode				
		BINHDXFF	X'80'	Half-duplex flip-flop mode				
		BINMSTSL	X'C0'	Simplex mode				
		BINRCVR	X'00'	Primary is responsible for recovery				
			X'20'	Sender of message is responsible for recovery				
Secondary LU Receive Pacing Count	BINRPACE			(Reserved)	8	8	8	8
Request Unit Sizes	BINSRUSZ			Maximum size of the request unit that can be sent by the secondary logical unit	9	9	9	9
	BINPRUSZ			Maximum size of the request unit that can be sent by the primary logical unit	10	A	10	A
Logical Unit Profile	BINLUP	BINLUP0C	X'00'	Logical unit profile 0	13	D	13	D
		BINLUP1C	X'01'	Logical unit profile 1				
		BINLUP2C	X'02'	Logical unit profile 2				
		BINLUP3C	X'03'	Logical unit profile 3				
Logical Unit Presentation Services	BINPSCHR			Logical unit presentation services (See Figures J-6 and J-7)	14	E	14	E
Primary LU Name Length	BINPRIML			(Reserved)	26	1A	26	1A
Primary LU Name	BINPRIM			(Reserved)	27	1B	27	1B
User Data Length	BINUSEL			Length of user data	35	23	35	23
User Data	BINUSE			User data	36	24	36	24

Figure J-5 (Part 2 of 2). The BNDAREA DSECT (ISTDBIND)

BNDAREA DSECT: BINPSCHR for Logical Unit Profile 1

Field	*DSECT DS or ORG label*	*DSECT EQU label*	*Value*	*Meaning (For EQI, meaning when bit setting is on or when byte value is set)*	*DOS/VS Displacement Dec*	*DOS/VS Displacement Hex*	*OS/VS Displacement Dec*	*OS/VS Displacement Hex*
FM Header Subset and Data Stream Profile	BINLUP1	BINFMS1C	X'10'	Function management header subset 1	14	E	14	E
		BINFMS2C	X'20'	Function management header subset 2				
		BINFMS3C	X'30'	Function management header subset 3				
	BINDSP1	BINDSP0C	X'00'	Data stream profile 0				
		BINDSP1C	X'01'	Data stream profile 1				
Primary Logical Unit FM Header Subset Flags	BINPFMB1	BINDESTS	X'80'	Transmission may be interrupted twice to send to another logical unit	15	F	F	F
		BINCMPCT	X'40'	Compacted data may be sent				
		BINPDIR	X'20'	PDIR may be sent				
		BINKDDSI	X'10'	Keyed direct data sets with Add or Erase Record headers using RECID headers may be sent				
		BINSDSI	X'08'	Sequential data sets with Add may be sent				
		BINSAI	X'04'	Sequentially accessed addressed direct data sets with Add, Note, and Note Reply in the opposite direction may be sent				
		BINSIDS	X'02'	Series IDs (with status in reply) are supported				
		BINARRR	X'01'	Add Replicate, Replace Replicate is supported				
	BINPFMB2	BINQDSI	X'40'	Query data sets are supported	16	10	16	10
		BINCSDS	X'20'	CREATE, SCRATCH, or SCRATCH ALL data sets are permitted				
		BINXFPD	X'10'	Execute FF delayed is not permitted				
Primary Logical Unit Data Stream Flags	BINPDSB1	BININTR	X'80'	Interactive data stream (BS, CR, LF, ENP, INP, HT, VT) may be sent	17	11	17	11
		BINHFDS	X'40'	Horizontal format data stream (SHF) may be sent				
		BINVTDS	X'20'	Vertical tab data stream (SVF) may be sent				
		BINVSDS	X'10'	Vertical select data stream (SVF (channel), SEL) may be sent				
		BINTRNDS	X'01'	Transparency data stream (TRN, IRS) may be sent				
Primary Logical Unit Media Flags	BINPMED1	BINDOCMT	X'80'	Document format may be sent	19	13	19	13
		BINCARD	X'40'	Card format may be sent				
		BINXCHNG	X'20'	Exchange media format may be sent				
		BINDISK	X'10'	Disk, data management format may be sent				
Secondary Logical Unit FM Header Subset Flags	BINSFMB1	BINDESTS	X'80'	Transmission may be interrupted twice to send to another logical unit	20	14	20	14
		BINCMPCT	X'40'	Compacted data may be sent				
		BINPDIR	X'20'	PDIR may be sent				
		BINKDDSI	X'10'	Keyed direct data sets with Add or Erase Record headers using RECID headers may be sent				
		BINSDSI	X'08'	Sequential data sets with Add may be sent				
		BINSAI	X'04'	Sequentially accessed addressed direct data sets with Add, Note, and Note Reply in the opposite direction may be sent				

Figure J-6 (Part 1 of 2). BINPSCHR Field of the BNDAREA DSECT for Logical Unit Profile 1

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
		BINSIDS	X'02'	Series IDs (with status in reply) are supported				
		BINARRR	X'01'	Add Replicate, Replace Replicate are supported				
	BINSFMB2	BINQDSI	X'40'	Query data sets are supported	21	15	21	15
		BINCSDS	X'20'	CREATE, SCRATCH, or SCRATCH ALL data sets are permitted				
		BINXFPD	X'10'	Execute FP delayed is not permitted				
Secondary Logical Unit Data Stream Flags	BINSDSB1	BININTR	X'80'	Interactive data stream (BS, CR, LF, ENP, INP, HT, VT) may be sent	22	16	22	16
		BINHFDS	X'40'	Horizontal format data stream (SHF may be sent				
		BINVTDS	X'20'	Vertical tab data stream (SVF) may be sent				
		BINVSDS	X'10'	Vertical select data stream (SVF (channel), SEL) may be sent				
		BINTRNDS	X'01'	Transparency data stream (TRN, IRS) may be sent				
Secondary Logical Unit Media Flags	BINSMED1	BINDOCMT	X'80'	Document format may be sent	24	18	24	18
		BINCARD	X'40'	Card format may be sent				
		BINXCHNG	X'20'	Exchange media format may be sent				
		BINDISK	X'10'	Disk, data mgmt format may be sent				

Figure J-6 (Part 2 of 2). BINPSCHR Field of the BNDAREA DSECT for Logical Unit Profile 1

BNDAREA DSECT: BINPSCHR for Logical Unit Profile 2

Field	DSECT DS or ORG label	DSECT EQU label	Value	Meaning (For EQU, meaning when bit setting is on or when byte value is set)	DOS/VS Displacement Dec	DOS/VS Displacement Hex	OS/VS Displacement Dec	OS/VS Displacement Hex
Presentation Space Size	BINPRESZ	BINPSZ1	X'01'	Character matrix 12 x 40	23	17	23	17
		BINPSZ2	X'02'	Character matrix 24 x 80				

Figure J-7. BINPSCHR Field of the BNDAREA DSECT for Logical Unit Profile 2

IBM-Supplied Session Parameters for Logical Units

IBM supplies session parameters for the various types of logical units that IBM supports. These parameters are provided in an IBM-supplied logon mode table (ISTINCLM for DOS/VS and OS/VS2 MVS or ISTINALM for OS/VS1 or OS/VS2 SVS). Figure J-8 shows the MODEENT macro instructions used in defining this table with the appropriate session parameters.

The session parameters in ISTINCLM or ISTINALM are used for a logical unit if you do not specify a logon mode table for that logical unit during VTAM definition. See the appropriate system programmer's guide for additional information. The way in which an application program can be written depends upon the protocols that these session parameters represent. The logical units for which there is an entry in the logon mode table are:

IBM 3270 Information Display System

IBM 3600 Finance System

IBM 3650 Retail Store System

IBM 3660 Supermarket System

IBM 3767 Communications Terminal

IBM 3770 Data Communications System

Some of the logical units listed in the table may accept logon mode names used by other logical units listed or may accept logon mode names not included in the IBM-supplied table. Logical units not listed may accept a logon mode name that is listed. Refer to the appropriate component description or programming guide for the logon modes acceptable to a particular logical unit.

```
ISTINCLM[1]  MODETAB
IBM3767      MODEENT  LOGMODE=INTERACT,FMPROF=X'03',TSPROF=X'03',PRIPROT=X'B1',SECPROT=X'A0',COMPROT=X'3040'
IBM3770      MODEENT  LOGMODE=BATCH,FMPROF=X'03',TSPROF=X'03',PRIPROT=X'A3',SECPROT=X'A3',COMPROT=X'7080'
IBMS3270     MODEENT  LOGMODE=S3270,FMPROF=X'02',TSPROF=X'02',PRIPROT=X'71',SECPROT=X'40',COMPROT=X'2000'
IBM3600      MODEENT  LOGMODE=IBM3600,FMPROF=X'04',TSPROF=X'04',PRIPROT=X'F1',SECPROT=X'F1',COMPROT=X'7000'
IBM3650I[2]  MODEENT  LOGMODE=INTRACT,FMPROF=X'00',TSPROF=X'04',PRIPROT=X'B1',SECPROT=X'90',COMPROT=X'6000'
IBM3650U[2]  MODEENT  LOGMODE=INTRUSER,FMPROF=X'00',TSPROF=X'04',PRIPROT=X'31',SECPROT=X'30',COMPROT=X'6000'
IBMS3650[2]  MODEENT  LOGMODE=IBMS3650,FMPROF=X'00',TSPROF=X'04',PRIPROT=X'B0',SECPROT=X'90',COMPROT=X'4000'
IBM3650P[2]  MODEENT  LOGMODE=PIPELINE,FMPROF=X'00',TSPROF=X'03',PRIPROT=X'30',SECPROT=X'10',COMPROT=X'0000'
IBM3660[2]   MODEENT  LOGMODE=SMAPPL,FMPROF=X'03',TSPROF=X'03',PRIPROT=X'A0',SECPROT=X'A0',COMPROT=X'0081'
IBM3660A[2]  MODEENT  LOGMODE=SMSNA100,FMPROF=X'00',TSPROF=X'00',PRIPROT=X'00',SECPROT=X'00',COMPROT=X'0000'
             MODEEND
```

Notes: [1] The name of this table in OS/VS1 is ISTINALM

[2] These logon mode names are not available in OS/VS2 SVS

Figure J-8. LOGMODE Entries in the IBM-Supplied Logon Mode Table (ISTINCLM)

GLOSSARY

This glossary defines terms and abbreviations that are important in this book. It does not include terms previously established for IBM operating systems and IBM products used with VTAM. Additional terms can be found by referring to the index, to prerequisite and corequisite books, and to the *IBM Data Processing Glossary*, GC20-1699.

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the *American National Standard Vocabulary for Information Processing* (Copyright ©1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of the American National Standards Committee X3. A complete commentary taken from ANSI is identified by an asterisk that appears between the term and the beginning of the commentary; a definition taken from ANSI is identified by an asterisk after the item number for that definition.

The symbol *ISO* at the beginning of a definition indicates that it has been discussed and agreed upon at meetings of the International Organization for Standardization Technical Committee 97/Subcommittee 1 (Data Processing), and has also been approved by ANSI.

The symbol *SC1* at the beginning of a definition indicates that it is reprinted from an early working document of ISO Technical Committee 97/Subcommittee 1 and that final agreement has not yet been reached among its participating members.

A

ACB. Access method control block.

accept. In VTAM, to connect a terminal to a VTAM application program as the result of a logon. The logon may be originated by the terminal, the network operator, another application program, or VTAM. Contrast with *acquire.*

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

acquire. In VTAM, to connect a terminal to a VTAM application program in the absence of a logon. The connection occurs at the program's initiative. Contrast with *accept.*

active. Pertaining to major node that is known to VTAM and is available for use or pertaining to a minor node that is connected to, or available for connection to, a VTAM application program. Contrast with *inactive.*

any-mode. In VTAM: (1) The form of a read or receive request that obtains data from one unspecified terminal. (2) The form of solicit request that solicits data from all eligible connected terminals. (3) The form of connection request that connects one unspecified terminal that has logged on. (4) Contrast with *specific-mode.* See also *continue-any mode.*

application program identification. The symbolic name by which a VTAM application program is identified to VTAM and other components of the network.

application program logon. Synonym for *simulated logon.*

APPLID routine. Synonym for *logon-interpret routine.*

asynchronous operation. In VTAM, an operation such as connection or data transfer in which the VTAM application program is allowed to continue execution while VTAM performs the operation. VTAM interrupts the program after the operation is completed.

asynchronous request. In VTAM, a request for an asynchronous operation.

automatic logon. In VTAM, a logon performed by VTAM on behalf of a terminal and a VTAM application program without either of them having to request it. The capability to log on automatically is specified during VTAM definition and can be modified by the network operator.

available. In VTAM: (1) Pertaining to a terminal that is active, is not connected to an application program, and for which there is no pending logon. (2) Pertaining to an exit routine that has been specified by an application program and that is not being executed.

B

basic mode. In VTAM, a mode of data transfer in which the VTAM application program can communicate with local 3270 Information Display Systems, start-stop terminals, and BSC terminals. Contrast with *record mode.*

block. In the basic mode of VTAM, a unit of data that is transmitted between a VTAM application program and a terminal.

bracket. In VTAM, an uninterruptible unit of work, consisting of one or more chains of request units and their responses, exchanged between an application program and a terminal. Examples are data base inquires/responses, update transactions, remote job entry output sequences to work stations, and similar applications.

bracket protocol. In SNA, a data flow control protocol in which exchanges between logical units (LUs) are achieved through the use of brackets, with one LU designated at session initiation as the first speaker, and the other LU as the bidder. The bracket protocol involves bracket initiation and termination rules.

C

change-direction protocol. In VTAM, a method of communication in which the sender stops sending on its own initiative, signals this fact to the receiver, and prepares to receive.

character-coded. In VTAM, pertaining to a logon or logoff command usually entered by a terminal operator from a keyboard and sent by a logical unit in character (unformatted) form. Contrast with *field-formatted.*

CID. Communication identifier.

closedown. The deactivation of a device, program, or system. See also *quick closedown, orderly closedown.*

cluster controller. See *cluster control unit* and *SDLC cluster controller.*

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 computer only through 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 Communication 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.*

command. (1) A request from a terminal for the performance of an operation or the execution of a particular program. (2) In SNA, a request unit initiating an action or beginning a protocol; it is used in contrast with reply, which is a request unit (not a response) that is sent in reaction to a command. For example: Quiesce (a data flow control request), which is a command, while Quiesce Complete is the reply.

communication control character. *A control character intended to control or facilitate transmission of data over communication networks.

communication identifier (CID). In VTAM, a key for locating the control blocks that represent an active session. The key, which consists of a pair of network addresses, is created when the session begins and deleted when the session ends.

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.

connection. In VTAM, the linking of VTAM control blocks in such a way that a VTAM application program can communicate with a terminal. Connection includes establishing and preparing the network path between the program and the terminal. See also *queued for connection.*

continue-any mode. In VTAM, a state into which a terminal is placed that allows its input to satisfy an input request issued in any-mode. While this state exists, input from the terminal can also satisfy input requests issued in specific-mode. Contrast with *continue-specific mode.*

continue-specific mode. In VTAM, a state into which a terminal is placed that allows its input to satisfy only input requests issued in specific-mode.

conversational write operation. In the basic mode of VTAM, an operation wherein data is first sent to a terminal and data is then read from that terminal.

D

data flow. In SNA, any of four flows in a given session, characterized as either primary-to-secondary or secondary-to-primary, and either normal or expedited.

data flow control. In SNA, a set of protocols and control functions within function management used to assist in controlling the flow of requests and responses within a session.

data transfer. In data communication, the sending of data from one point in a communication network and the receiving of the data at another point in the network.

data transmission. The sending of data from one point in a communication network for reception elsewhere by means of a channel or communication line.

definite response. In SNA, a form of response requested in the request header for a request unit; the receiver is requested to return a response whether positive or negative. Contrast with *exception response.*

device control character. (ISO) A control character used for the control of ancillary devices associated with a data processing system or data communication system, for example, for switching such devices on or off.

disconnection. In VTAM, the dissociation of VTAM control blocks in such a way as to end a session between a VTAM application program and a connected terminal. The disconnection process includes suspending the use of the network path between the program and the terminal.

E

error lock. In the basic mode of VTAM, a condition in which communication with a BSC, start-stop, or local 3270 terminal is suspended until a reset operation occurs.

exception message. In communicating with a logical unit, a message that indicates an unusual condition such as a sequence number being skipped. When VTAM detects such a condition, it notifies the VTAM application program. VTAM and/or the VTAM application program provides sense information which is included in the response that is sent to the logical unit.

exception response. (1) In SNA, a response requested in the RH for a request unit; the receiver is requested to return a response only if it is negative. Contrast with *definite response.* (2) Synonym for *negative response.*

exit list (EXLST). In VSAM or VTAM, a control block that contains the addresses of user-written routines that receive control when specified events occur during execution; for example, routines that process logons or I/O errors.

exit routine. In VTAM, any of several types of special-purpose user-written routines. See *accounting exit routine, authorization exit routine, EXLST exit routine, logon-interpret routine,* and *RPL exit routine.*

EXLST exit routine. In VTAM, a user-written routine whose address has been placed in an exit list (EXLST) control block. See also *RPL exit routine.*

expedited flow. In SNA, a data flow that is independent of and controls the normal flow. Data flow is split into normal and expedited flows. Requests and responses on a given (normal or expedited) flow are processed sequentially within the path, but the expedited flow traffic may be moved ahead of the normal flow traffic within the path. Contrast with *normal flow.*

H

host computer. (1) The primary or controlling computer in a multiple computer operation. (2) A computer used to prepare programs for use on another computer or on another data processing system; for example, a computer used to compile, link-edit, or test programs to be used on another system. (3) The primary or controlling computer in a data communication system. (4) In a VTAM telecommunication system, the processing unit in which VTAM resides.

I

inactive. In VTAM, pertaining to a major node that is unknown to VTAM and is unavailable for use, or pertaining to a minor node that is not connected to nor available for connection to a VTAM application program. Contrast with *active.*

interpret table. In VTAM, a user-defined correlation list that translates an argument into a string of eight characters. Interpret tables can be used to translate logon data into the name of an application program for which the logon is intended.

L

LDO. Logical device order.

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.

line. See *communication line.*

line control. The scheme of operating procedures and control signals by which a telecommunication system is controlled.

line discipline. Synonym for *protocol.*

line group. One or more communication lines of the same type that can be activated and deactivated as a unit.

local. (1)Pertaining to the attachment of devices directly by I/O channels to a host computer. Contrast with *remote.* (2) In data communication, pertaining to devices that are attached to a controlling unit by cables, rather than by data links.

logical device order (LDO). In VTAM, a set of parameters that specify a data-transfer or data-control operation to local 3270 Information Display Systems and certain kinds of start/stop or BSC terminals.

logical error. In VTAM, an error condition that results from an invalid request; a program logic error.

logical unit. In SNA, one of three types of network addressable units (NAUs). It is the port through which an end user accesses function management in order to communicate with another end user. It is also the port through which the end user accesses the services provided by the system services control point (SSCP). It must be capable of supporting at least two sessions – one with the SSCP, and one with another logical unit. It may be capable of supporting many sessions with other logical units. See also *physical unit, system services control point.*

log off. In VTAM, to request that a terminal be disconnected from a VTAM application program.

logoff. In VTAM, a request that a terminal be disconnected from a VTAM application program.

log on. In VTAM, to request that a terminal be connected to a VTAM application program.

logon. In VTAM, a request that a terminal be connected to a VTAM application program. See also *automatic logon* and *simulated logon.*

logon data. In VTAM: (1) The data portion of a field-formatted or character-coded logon from an SNA terminal. (2) The entire logon sequence from a non-SNA terminal. (3) Synonymous with *logon message.*

logon-interpret routine. In VTAM, a user-written routine that translates logon data. It may also verify the logon. Synonymous with *APPLID routine.*

logon message. Synonym for *logon data.*

logon mode. In VTAM, the communication protocols that govern a session between a logical unit and a VTAM application program. Synonymous with *session parameters.*

logon mode name. In VTAM, the symbolic representation used by a VTAM application program and a logical unit to refer to a logon mode.

logon mode table. In VTAM, a set of macro-generated constants making up one or more logon modes. Each logon mode is associated with a logon mode name.

M

major node. In VTAM, a set of minor nodes that can be activated and deactivated as a group. See also *minor node.*

message. (1) *An arbitrary amount of information whose beginning and end are defined or implied. (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. (3) (SC1) A sequence of characters used to convey data. The sequence usually consists of three parts: the heading, the text, and one or more characters used for control or error-detection purposes. (4) In telecommunications, a combination of characters and symbols transmitted from one point to another.

minor node. In VTAM, an element of the telecommunication network that can be activated or deactivated by the VARY command. See also *major node.*

multithread application program. A VTAM application program that processes many requests from many terminals concurrently. Contrast with *single-thread application program.*

N

NCP. Network control program.

negative response. A response indicating that a message did not arrive successfully or is unacceptable. Synonymous with *exception response.* Contrast with *positive response.*

negative response to polling limit. For a start-stop or BSC terminal, the maximum number of consecutive negative responses to polling that the communications controller accepts before suspending polling operations.

network. (1) (SC1) The assembly of equipment through which connections are made between terminal installations. (2) In data communication, a configuration in which two or more terminal installations are connected.

network control program (NCP). A program, generated by the user from a library of IBM-supplied modules, that controls the operation of the communications controller.

network control program generation. The process, performed in a host system, of assembling and link-editing a macro instruction program to produce a network control program.

network definition. In VTAM, the process of defining the identities and characteristics of each node in the telecommunication system and the arrangement of the nodes in that system.

network operator. (1) The person responsible for controlling the operation of a telecommunication network. (2) A VTAM application program authorized to issue network operator commands.

network operator command. A command used to monitor or control the telecommunication network.

network operator logon. A logon requested on behalf of a terminal by means of a network operator command.

NIB. Node initialization block.

NIB list. A series of contiguous node initialization blocks.

node. (1) An addressable point in a data communication network. (2) In VTAM, a point in a telecommunication system defined by a symbolic name. See also *major node* and *minor node.*

node initialization block (NIB). In VTAM, a control block associated with a particular terminal that contains information used by the VTAM application program to identify the terminal and indicate how communication requests directed at the terminal are to be processed.

node name. In VTAM, the symbolic name assigned to a specific major or minor node during network definition.

non-SNA terminal. In VTAM, a terminal that is part of a local 3270 Information Display System or a terminal supported by VTAM that uses start-stop or BSC protocol.

normal flow. In SNA, a data flow that is used for most requests and responses. Data flow is split into normal and expedited flows. The expedited flow is independent of and used to control the normal flow. Requests and responses on a given (normal or expedited) flow are processed sequentially within the path, but the expedited flow traffic may be moved ahead of the normal flow traffic within the path. Contrast with *expedited flow.*

O

orderly closedown. The orderly deactivation of VTAM and the telecommunication network. An orderly closedown does not take effect until all application programs have been disconnected from VTAM. Until then, all data transfer operations continue. Contrast with *quick closedown.*

P

path. (1) In VTAM, the intervening nodes and data links connecting a terminal and an application program in the host computer. (2) In SNA, the series of nodes, data links, and common network components (path control and data link control) that form the complete route traversed by the information exchanged between two network addressable units in session.

physical unit. (1) The control unit or cluster controller of an SNA terminal. (2) The part of the control unit or cluster controller that fulfills the role of a physical unit as defined by systems network architecture.

positive response. A response that indicates a message was received successfully. Contrast with *negative response.*

program operator. A VTAM application program that is authorized to issue network operator commands.

protocol. (1) In SNA, the sequencing rules for requests and responses by which network addressable units in a communication network coordinate and control data transfer operations and other operations. See also *bracket protocol.* (2) Synonymous with *line discipline.*

Q

queued for connection. In VTAM, the state of a terminal that has logged on to an application program but has not yet been accepted by that application program. See also *connection.*

quick closedown. In VTAM, a closedown in which current data-transfer operations are completed, while new connection and data-transfer requests are canceled. Contrast with *orderly closedown.*

quiesce protocol. In VTAM, a method of communicating in one direction at a time. Either the VTAM application program or the logical unit assumes the exclusive right to send normal-flow messages, and the other node refrains from sending such messages. When the sender wants to receive, it releases the other node from its quiesced state.

R

RDT. Resource definition table.

record mode. In VTAM, a mode of data transfer in which the application program can communicate with logical units or with local or remote 3270 Information Display Systems. Contrast with *basic mode.*

remote. In data communication, pertaining to devices that are connected to a data processing system through a data link.

reply. In SNA, a request unit sent in reaction to a previously received request unit (command). Contrast with *response.* See also *command.*

request. (1) A directive that causes a data transfer or related operation to be performed. (2) In SNA, synonym for *request unit.*

request header. In SNA, a request/response header that indicates a request.

request parameter list (RPL). In VTAM, a control block that contains the parameters necessary for processing a request for data transfer, for connecting or disconnecting a terminal, or for some other operation.

request/response unit (RU). In SNA, the basic unit of information entering and exiting the transmission subsystem. It may contain data, acknowledgment of data, commands that control the flow of data through the network, or responses to commands.

resource definition table (RDT). In VTAM, a table that describes the characteristics of each node available to VTAM and associates each node with an address.

responded output. In VTAM, a type of output request that is completed when a logical unit receives a message and returns a response (if one is called for). Contrast with *scheduled output.*

response. (1) An answer to an inquiry. (2) The unit of information that is exchanged between VTAM or a VTAM application program and an SNA terminal to describe how a message arrived. (3) In SNA, synonym for *response unit.* (4) Contrast with *command, reply.*

RPL. Request parameter list.

RPL-based macro instruction. In VTAM, a macro instruction whose parameters are specified by the user in a request parameter list.

RPL exit routine. In VTAM, a user-written routine whose address has been placed in the EXIT field of a request parameter list. VTAM invokes the routine to indicate that an asynchronous request has been completed. See also *EXLST exit routine.*

RU. Request/response unit.

S

scheduled output. In VTAM, a type of output request that is completed, as far as the application program is concerned, when the program's output data area is free. Contrast with *responded output.*

SDLC. Synchronous data link control.

SDLC cluster controller. A cluster control unit for a teleprocessing subsystem.

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 unless reset by the application program.

session. (1) The period of time during which a user of a terminal can communicate with an interactive system; usually, the elapsed time from when a terminal user logs on the system until he logs off the system. (2) The period of time during which programs or devices can communicate with each other. (3) In SNA, a logical connection, established between two network addressable units (NAUs), that allows them to communicate. The session is uniquely identified by a pair of network addresses, identifying the origin and destination NAUs of any transmissions exchanged during the session. See *LU-LU session, SSCP-LU session, SSCP-PU session.*

session control. In SNA, one of the components of transmission control. It is responsible for allocating resources necessary for a session, for purging data flowing in a session if an unrecoverable error occurs, and for resynchronizing the data flow after such an error.

session parameters. Synonym for *logon mode.*

shared. Pertaining to the availability of a resource to more than one user at the same time.

simulated logon. A logon generated for a terminal by VTAM at the application program's request. The application program accepts or rejects the terminal as if it had logged on. Synonymous with *application program logon.*

single-thread application program. A VTAM application program that processes requests from terminals one at a time. Such a program usually requests synchronous operations from VTAM, waiting until each operation is completed before proceeding. Contrast with *multithread application program.*

SNA. Systems network architecture.

SNA terminal. In VTAM: (1) A physical unit or logical unit. (2) A terminal that is compatible with systems network architecture.

solicit. In VTAM, to obtain data from a BSC or start-stop terminal or from a local 3270 terminal and move the data into VTAM buffers.

solicited message. A response from VTAM to a network operator command entered by a program operator. Contrast with *unsolicited message.*

specific-mode. In VTAM: (1) The form of read, receive, or solicit request that obtains data from one specific terminal. (2) The form of connection request that connects a specific terminal that has logged on. (3) Contrast with *any-mode.* See also *continue-specific mode.*

synchronous operation. In VTAM, a connection, communication, or other operation in which VTAM, after receiving the request for the operation, does not return control to the program until the operation is completed. Contrast with *asynchronous operation.*

synchronous request. In VTAM, a request for a synchronous operation. Contrast with *asynchronous request.*

systems network architecture (SNA). The total description of the logical structure, formats, protocols, and operational sequences for transmitting information units through the communication system. Communication system functions are separated into three discrete areas: the application layer, the function management layer, and the transmission subsystem layer. The structure of SNA allows the ultimate origins and destinations of information—that is, the end users—to be independent of, and unaffected by, the specific communication-system services and facilities used for information exchange.

T

telecommunication network. In a telecommunication system, the combination of all terminals and other telecommunication devices and the data links that connect them.

telecommunication system. In a teleprocessing system, those devices and functions concerned with the transmission of data between the data processing system and the remote users.

teleprocessing subsystem. In VTAM, a secondary or subordinate network and set of programs that are part of a larger teleprocessing system; for example, the combination consisting of an SDLC cluster controller, its stored programs, and its attached terminals.

teleprocessing system. A data processing system in combination with data communication facilities.

terminal. (1) A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a communication channel. (2) In VTAM, an end point in a telecommunication network; that is, a physical or logical unit, a start-stop or BSC device, or a 3270 Information Display System.

terminal component. A separately addressable part of a terminal that performs an input or output function, such as the display component of a keyboard-display device.

transmission. In data communication, one or more blocks or messages. For BSC and start-stop devices, a transmission is terminated by an EOT character. See also *block* and *message.*

transmission control unit (TCU). A communication control unit whose operations are controlled solely by programmed instructions from the computing system to which the unit is attached; no program is stored or executed in the unit. Contrast with *communications controller.*

transparent text mode. A mode of binary synchronous transmission in which only transmission control characters preceded by DLE are acted upon as line control characters. All other bit patterns that happen to be transmission control characters are transmitted as data.

U

unsolicited message. A network operator message, from VTAM to a program operator, that is unrelated to any command entered by the program operator. Contrast with *solicited message.*

V

Virtual Telecommunications Access Method (VTAM). A set of programs that control communication between terminals and application programs running under DOS/VS, OS/VS1, and OS/VS2.

VTAM. Virtual Telecommunications Access Method.

VTAM application program. Any program that uses VTAM macro instructions.

VTAM definition. The process of defining the communication network to VTAM and modifying IBM-defined VTAM characteristics to suit the needs of the user.

INDEX

GC27-6995-5

VTAM Macro Language Reference (File No. S370-30) Printed in U.S.A. GC27-6995-5

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601