GC27-6995-0
File No. S370-30

Systems

VTAM Application Programmer's Reference Manual

Virtual Telecommunications Access Methods (VTAM)

For Planning Purposes Only

IBM

First Edition (April 1973)

The material in this manual is preliminary and should be used for planning purposes only. Changes will be made to the information herein and will be reported in subsequent revisions or technical newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form has been provided at the back of this publication for reader's comments. If the form has been removed, address comments to the IBM Corporation, Department 643, Neighborhood Road, Kingston, N.Y. 12401

Preface

This book is a catalog of the macro instructions that allow new programs to use the Virtual Telecommunications Access Method (VTAM). It provides those who will be coding such programs an opportunity to study the form and scope of these macro instructions before VTAM becomes available.

With few exceptions, VTAM macro instructions can be coded without regard for the particular operating system (DOS/VS, OS/VS1, or OS/VS2) under which the program will be running. Those exceptions are clearly identified in the macro instruction descriptions.

The first section of this book explains in a general way the services provided by VTAM for the new program, and indicates the macro instructions that are used to request them.

The second section contains detailed descriptions of each macro instruction, arranged in alphabetic order. Each description is presented in a fixed format, and standard conventions are used throughout to indicate how the macro instructions are to be coded. The section begins with an explanation of these conventions and an illustration showing how the macro instruction descriptions are arranged.

The appendixes indicate line control characters operated on by VTAM, and describe macro completion information provided by VTAM. 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 must be familiar with *Introduction to VTAM* and with those parts of the *OS/VS and DOS/VS Assembler Language* 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-control discipline (start- stop or binary synchronous) that will be used with each one, and with teleprocessing concepts in general.

Note: This edition of the VTAM Language Reference Manual is to be used for *planning purposes only.* This means that any portion of this book's content is subject to change.

Contents

Figures

Part I: The VTAM Language

The Virtual Telecommunications Access Method (VTAM) provides a program running under a virtual storage operating system with the ability to communicate with the terminals of a telecommunications network. The VTAM language described in this book is the set of macro instructions that are available to request this communication.

VTAM provides a mechanism by which a program can read, write, or perform other I/O operations with a terminal. As the next section shows, however, the communication provided by VTAM involves more than the straightforward transfer of data between a program and a terminal.

What exactly then does the VTAM language provide?

Services Provided Through the VTAM Language

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.

Obtain data from one or a group of terminals, and move the data into VTAM buffers. Repeat this action until a specified amount of data has been received (SOLICIT).

Move data from VTAM buffers to an area in program storage – either using data previously obtained from a specified terminal or using data previously obtained from any terminal (READ).

Obtain data from a specific terminal and move it directly into an area in program storage (READ).

Move data from an area in program storage to a specified terminal (WRITE).

Automatically follow an output operation with an input operation (WRITE).

Erase all or the unprotected portion of a 3270 display unit (WRITE).

Read the entire contents of a 3270 display unit buffer (DO, LDO).

Copy the contents of a remotely attached 3270 display unit buffer into the buffer of another remotely attached display unit or printer (DO, LDO).

Send a negative or positive acknowledgement accompanied by leading graphic characters to a 3735 terminal or to a System/3 or System/370 CPU (DO, LDO).

Cancel an I/O operation prematurely; reset an error lock set for a device (RESET).

Check the completion status of any of the above activities (CHECK).

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, then by another, connection between the program and the terminal must be established before any I/O activity can take place. The connection operation 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 describes the parameters of 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 will affect subsequent communication with that terminal (NIB).

Change the information in the above control block, once the control block is in use (CHANGE).

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).

Open the control block that identifies the program to VTAM and the telecommunications network, and optionally allow terminals to request connection to the program (OPEN).

Establish connection with a terminal or with a group of terminals (OPNDST).

Simulate a terminal's request for connection, so that a user-written routine that handles such requests will be invoked (SIMLOGON).

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 or the logon message of a terminal requesting connection, find out how many terminals are currently connected to the program and how many are waiting to become connected, or determine the availability of other application programs (INQUIRE).

Translate a terminal's logon message into the name of an application program, as defined by the installation (INTRPRET).

Disconnect a terminal from the program; optionally request that the disconnected terminal be connected to another program (CLSDST).

Disconnect all terminals from the program, and close the control block that is used to identify the program to VTAM and the telecommunications network (CLOSE).

How the VTAM Language Is Used

There is no single, predetermined way to code a program that uses VTAM. Although there are many different ways that the program can be written, four things are true of all such programs:

1. Every program must lay the groundwork for communication with its terminals. This *preparation* deals exclusively with the building of control blocks and the setting of various fields within them. All control blocks need not be built at one time. Preparation includes the subsequent manipulation of these control block fields.

2. Every program must activate a control block that identifies the program to VTAM and the teleprocessing network before the following steps can be performed. This activity involves *opening an access method control block.*
3. Every program must request *connection* between itself and a terminal before it can communicate with that terminal.
4. When the above actions have been completed, the program can engage in *communication* with its connected terminals.

Preparing Control Blocks

There are four VTAM control blocks that are used by the program during opening, connection, and communication.

Access Method Control Block (ACB). Serves as a link between VTAM and the connection and communication requests that refer to it, thus providing the program access to the VTAM programs that will process these requests. The ACB can also provide terminals throughout the teleprocessing network with access to the program. To a terminal user, the ACB in effect represents the application program with which it interacts. A program can use several ACBs, but from the point of view of the terminal user, each ACB is a different application program.

Exit List (EXLST). Contains the addresses of user-written routines that are to receive control when certain types of events occur, such as I/O errors, logon requests, and attention interruptions.

Request Parameter List (RPL). Describes the characteristics of a desired connection or communication operation. Since the RPL contains information related to only one specific request, the program will likely modify a given RPL many times or establish many different RPLs.

Node Initialization Block (NIB). Identifies a terminal and provides various parameters that govern all communication requests directed at that terminal.

ACB, EXLST, RPL, and NIB macro instructions are used to build their respective control blocks during program assembly. A GENCB macro instruction is available to generate any of these four control blocks during program execution – optionally in dynamically allocated storage.

Most of the fields in these control blocks are set by the program to tell VTAM how to perform an action; others are set by VTAM to inform the application program what happened when the action was performed. Some work both ways.

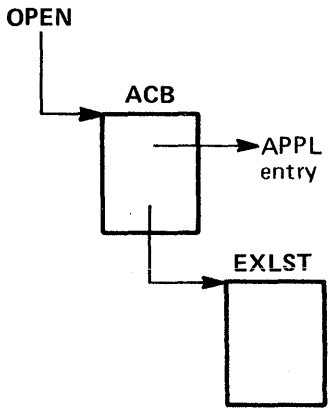
The displacements of each field within each control block need not concern the programmer, since a set of macro instructions are provided to manipulate any of the fields. The way these macro instructions (SHOWCB, TESTCB, and MODCB) are used to manipulate a field is quite similar to the way the ACB, EXLST, RPL, NIB, and GENCB macro instructions are used to set the fields of the control block when it is built. An example illustrates not only what this similarity is, but also illustrates an important concept of field naming:

> If AREALEN=100 is specified in an RPL macro instruction (or a GENCB macro instruction), an RPL is built and a particular field within it is set to the value 100. That field is called the AREALEN field throughout all VTAM documentation.
>
> If FIELDS=AREALEN is coded in a SHOWCB macro instruction, the content of this same field is moved to a designated area in program storage. Coding AREALEN=100 in a TESTCB macro instruction causes the contents of this

field to be compared with the value 100 and the PSW condition code to be set accordingly. And if AREALEN=100 is coded in a MODCB macro instruction, the value 100 is set in that field.

So: The *name* of any control block field is synonymous with the *keyword* (the characters to the left of the equal sign) of the operand used to build that field. It will be useful to remember this principle when using the macro instruction descriptions in this book.

Opening the ACB



This step requires an ACB (and usually an exit list); it is implemented with the OPEN macro instruction.

Before an application program can use VTAM, an entry representing that program must be included in the resource definition table during VTAM definition (this entry is generated by the installation). Later, when the program is running, it must indicate to VTAM that it is the program represented by that entry. The program does so by creating and opening an ACB that indicates the APPL entry.

If the installation includes a password when it defines the APPL entry, the program must include that password in the ACB being opened.

Terminal users who want to become connected to the application program must cite the APPL entry name (or what amounts to an alias of it, as defined by the installation in a logon characteristics table) in their logon request. If the ACB associated with that name has been successfully opened, the application program represented in the logon request is active.

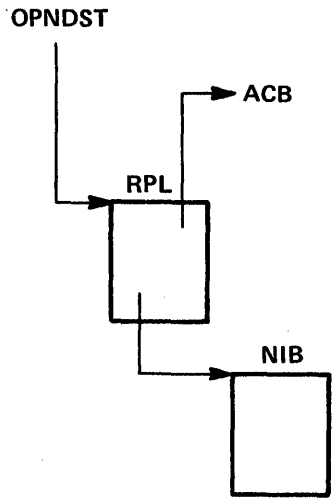
If the program is to handle logon requests in a routine that will automatically be invoked when the request occurs, the ACB must point to an exit list (EXLST). The exit list must in turn indicate the address of this routine. (The exit list can indicate a variety of other routines as well – routines that will be invoked when errors or other special conditions arise during program execution.) The task of this routine (called the LOGON exit list routine) is to evaluate the logon request to determine whether connection should be established between the program and the terminal, and to request VTAM to perform that connection. OPEN not only allows terminals to commence issuing logon requests, but can also cause VTAM to issue logon requests on behalf of certain terminals. (During VTAM definition, the installation can indicate that it wants VTAM to automatically generate logon requests for specified terminals when the application opens its ACB.)

When a program is finished processing, it closes the ACB with a CLOSE macro instruction. CLOSE causes any connected terminals to be disconnected. Once the ACB has been closed, the ACB and the portion of the program represented by it ceases to exist as far as the rest of the telecommunications network is concerned.

All of the activity associated with opening an ACB is performed solely for the purpose of preparing for connection.

Connecting Terminals

Connection requires an RPL and a NIB, in addition to the ACB required for opening. Connection is requested with the OPNDST (open destination) macro instruction.



When OPNDST is issued, an RPL is indicated that in turn points to a NIB. Depending on the type of OPNDST used, the NIB may in turn indicate the symbolic name of the terminal that is to be connected.

There are two different ways to establish connection:

1. The application program takes the initiative and establishes connection with a specified terminal. Here the terminal's symbolic name is known to the application program. This process is called *acquiring* a terminal.

   The application program establishes a NIB (node initialization block) in which it identifies the terminal's symbolic name and indicates along with it a variety of options that will affect I/O operations performed with that terminal. An OPNDST macro instruction is issued, and the terminal is connected.

   An option of the NIB macro instruction allows the programmer to group NIBs together so that a request to connect the first in the group is interpreted by VTAM as a request to connect all. These groups are called *NIB lists.*

2. The application program can let terminals take the initiative and establish connection only when terminals issue logon requests or when VTAM issues logon requests on behalf of a terminal. This process is called *accepting* a terminal. This activity can take place in a LOGON exit list routine, invoked after the ACB has been opened. (A terminal cannot directly ask VTAM to connect it to the program. The terminal can only ask that the program in turn request VTAM to connect the terminal and the program.)

   When the LOGON exit list routine is invoked, it is supplied with the symbolic name of the terminal. Using this name, the routine can determine whether or not connection should be established (that is whether or not OPNDST should be issued).

The application program can employ several macro instructions that do not directly request connection, but instead request operations that are related to connection:

The INQUIRE macro instruction can be used to obtain information about terminals that are either already connected, or have issued logon requests and are candidates for connection. This information includes logon messages and device characteristics.

The SIMLOGON macro instruction can be employed to use the LOGON exit list routine to connect terminals that have not issued logon requests. SIMLOGON generates a logon request as though the terminal had issued one, which causes the LOGON exit list routine to be invoked. This makes SIMLOGON essentially a macro instruction that allows the application program to use its terminal-accepting mechanism (LOGON exit list routine) for the purpose of *acquiring* terminals.

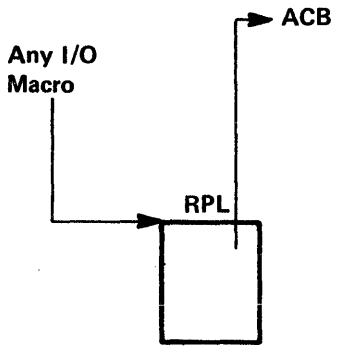
As long as terminals continue to generate logon requests, VTAM queues these requests until they are satisfied by the program. The SETLOGON macro instruction causes VTAM to stop queuing new logon requests. OS/VS1 and OS/VS2 users can use a variation of SETLOGON to resume the queuing of logon requests.

The CHECK macro instruction either causes program execution to stop until a connection is completed, or it causes user-written routines to be invoked when a connection operation is completed unsuccessfully.

Disconnection is provided by the CLSDST (close destination) macro instruction. An option of CLSDST allows the program to disconnect a terminal, generate a logon request for the terminal, and send that request on to another program. The effect of this option is to pass a terminal from one program to another.

Connection, and all the activity associated with it, is simply the prerequisite for communication.

Communicating with Terminals



Communication requires an RPL, in addition to the ACB that is required for connection. Communication – the performance of I/O and I/O-related operations between the program and a terminal – is requested via a set of macro instructions. Each macro instruction points to an RPL, which contains fields describing the exact nature of the operation to be performed and the identity of the terminal with which it is to be performed.

VTAM allows the program to both transfer data back and forth between itself and a terminal and perform related I/O operations like erasing terminal buffers.

The programmer must take into account the physical characteristics of the terminal with which his program is communicating. Specifically, anyone coding programs for VTAM must be aware of:

All the device control characters that will come from, or that must be sent to, each terminal.

The length of the data that will be received from, or can be sent to, each terminal.

The functions available in each terminal, so that actions that cannot be accomplished by the terminal will not be requested or expected.

This kind of information can be found in the component description manuals for each terminal. A list of the devices supported by VTAM is contained in Figure 4 located in the description of the NIB macro instruction.

The bulk of a typical application program's communication requests will likely consist of these three macro instructions:

SOLICIT, READ, and WRITE.

The SOLICIT macro instruction initiates the polling and data transfer activity required to obtain data from a terminal, as well as the returning of responses required to repeat the process until a designated amount of data has been received. The data is stored in VTAM buffers where it is available for transfer to program storage by the READ macro instruction.

Data can be solicited from a specific terminal, or it can be solicited from all connected terminals. After a solicited terminal responds, and while the application program proceeds to read from and write to it, SOLICIT continues to solicit data from other terminals.

RPL fields govern the identity of the terminals; fields set in each terminal's NIB during connection govern the extent of the solicitation to be performed – that is, whether SOLICIT obtains a block of data, a message, or a transmission.

The READ macro instruction causes data in VTAM buffers (brought there as a result of a previous solicit operation) to be transferred to a designated storage area

in the program. READ can be used to obtain data from any terminal from which data has already been solicited, or it can be used to obtain data from a specified terminal. When the latter is done and no data has yet been solicited from that terminal, VTAM causes a solicit operation to be performed first. Fields set in the RPL establish the identity of the terminal from which the data is to be read and indicate the location in the program where the data is to be placed.

The WRITE macro instruction writes a block of data from program storage to a specific terminal. Fields set in the RPL identify the terminal, the location and amount of data to be sent, and indicate whether end-of-block, end-of-message, or end-of-transmission line control characters are to be inserted. Other RPL fields implement optional variations of WRITE, such as erasing the screen of an alphameric display device or automatically following the write operation with a read operation.

Figure 1 provides a simple example of how a program might use SOLICIT, READ, and WRITE macro instructions.

The application program can employ several macro instructions that do not directly request communication, but instead request operations that are related to communication:

The INQUIRE macro instruction obtains device characteristics of connected terminals. The program can use this information to set certain options in that terminal's NIB; this establishes the proper options for all communication with that terminal when OPNDST is issued.

Should the application program alter the information represented in a terminal's NIB after OPNDST has been issued, the CHANGE macro instruction is used to make these alterations effective.

The RESET macro instruction cancels pending I/O operations, cancels I/O operations that are in progress, or resets error locks set for a specific terminal.

The CHECK macro instruction can be used to either stop program execution until a given I/O operation is completed, or to invoke user-written routines if the I/O operation is completed with an error. (CHECK also supports connection requests.)

DO and LDO

By using DO and LDO macro instructions, the application program can perform certain specific I/O operations like copying the contents of a 3270 display unit buffer to a 3270 printer buffer, or reading the contents of a 3270 display unit before the terminal operator presses the ENTER function key.

To request these operations, the VTAM user creates a control block called an LDO (logical device order) with the LDO macro instruction, and then issues a DO macro instruction to initiate the operation indicated in the LDO. (The DO macro instruction points to an RPL, which in turn must be set to point to the LDO.)

LDOs can be grouped together to indicate a sequence of operations to be performed. In both form and manner of use, LDOs resemble the channel commands words of a channel program.

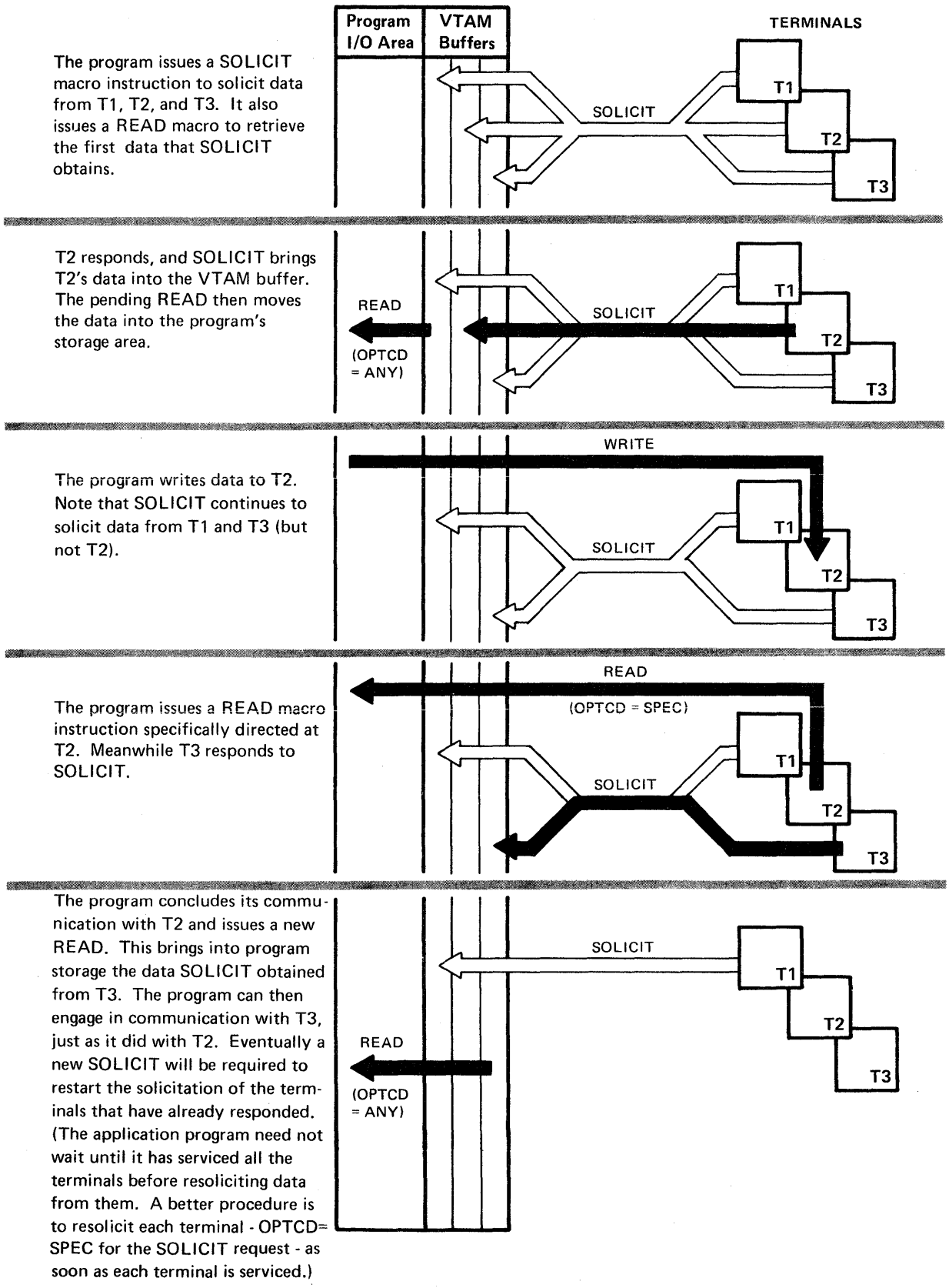


Figure 1. Using SOLICIT, READ, and WRITE to Communicate with a Terminal

The part of the LDO that indicates the operation to be performed is called the LDO command. The LDO commands are shown below, with descriptions of the actions they represent. The other parts of an LDO are a pointer to data to be sent (or data area where data is to be received), an indication of the length of the data or data area, and an indicator that can chain the LDO to another LDO.

COPYLBM
Copies the contents of a 3270 display unit buffer into the buffer of any printer or display unit attached to the same control unit. VTAM sends the data as a message (that is, VTAM ends the data with an ETX line control character).

COPYLBT
Operates like COPYLBM, except that VTAM sends the data as a transmission (that is, VTAM ends the data with an ETX character, waits for an acknowledgement, and then sends an EOT character).

READ
Obtains a block of data from a 3735 terminal or from a System/3 or System/370 CPU. The READ LDO obtains data in the same manner as the READ macro instruction. When combined with a WRTPRLG or WRTNRLG LDO, however, the application program can send acknowledgments to the device that include leading graphic characters.

READBUF
Obtains the entire contents of a 3270 display unit buffer and places it in program storage. (An ordinary READ macro instruction obtains data only after the terminal operator has pressed the ENTER key.)

WRITE
Sends a block of data to a 3735 terminal or to a System/3 or System/370 CPU. When WRITE is preceded by a WRTHDR LDO, VTAM prefixes the block of data with a user-supplied heading block.

WRITELBM
Operates like the WRITE LDO, except that the data is ended with an ETX character instead of an ETB character. A WRITELBM LDO can also be preceded by a WRTHDR LDO to include a heading block with the data.

WRITELBT
Operates like the WRITELBM LDO, except that VTAM waits for the device to acknowledge receipt of the data and then sends an EOT character. A WRITELBT LDO can also be preceded with a WRTHDR LDO to include a heading block with the data.

WRTHDR
Sends a block of user-provided heading characters to a 3735 terminal or to a System/3 or System/370 CPU. VTAM inserts an SOH character at the beginning, and an ETB character at the end. If a WRTHDR LDO is combined with a WRITE, WRITELBM, or WRITELBT LDO, however, the ETB character is not inserted, and the heading block is combined with the text block.

WRTNRLG
Sends a negative response (NAK) to a 3735 terminal or to a System/3 or System/370 CPU. The response can be accompanied by up to seven user-specified leading graphic characters. This LDO must be followed by a READ LDO (to obtain the data re-sent by the device).

WRTPRLG
Sends a positive response (ACK0 or ACK1) to a 3735 terminal or to a System/3 or System/370 CPU. As with the WRTNRLG LDO, the user can specify up to seven leading graphic characters to be sent with the response. The WRTPRLG LDO must also be followed by a READ LDO (to obtain the next block of data -- or EOT character -- sent by the device).

Part II: Macro Instructions

This part describes the macro instructions that enable a program to request the services described in Part 1. It begins by telling how these macro instructions are described.

How 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.

The 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 referred to as the macro instruction's assembler format table.

NAME: The macro instruction name provides a label for the macro instruction. It is usually associated with the first executable instruction of the macro expansion generated during program assembly. 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: A given macro instruction's routines generally are capable of performing many variations of the macro's basic function. It is through a macro instruction's operands that the user can tailor the macro's basic function to meet his specific needs. 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 keyword), an equals sign, and a value. The presence or absence of the equals sign provides a quick way to distinguish between keyword and positional operands. Keyword operands do not have to be coded in the order shown in the operands column. For example, a macro having an "AREALEN=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.

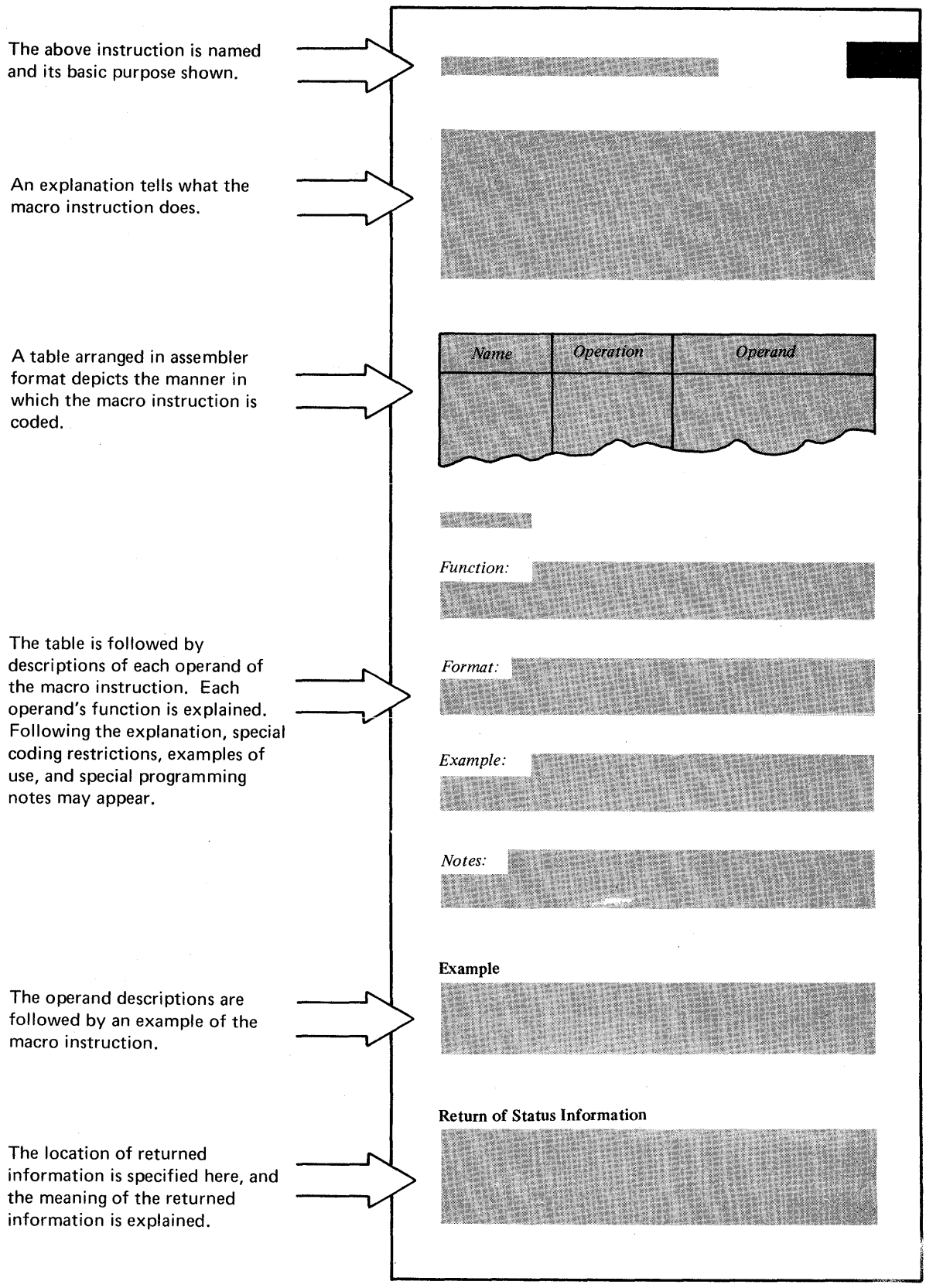


Figure 2. Macro Instruction Descriptions Follow a Fixed Arrangement

Positional operands must be coded in exactly the 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 A, B, and C. If all three are used, they are coded as...

A,B,C

but if only A and C are wanted, they are coded as...

A,,C

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

An operand field might contain both positional and keyword operands. In this case, all positional operands must precede any keyword operands:

B,C,AREALEN=132,AREA=WORKAREA

Operand Notation: A notational scheme is followed in the operands column to show how, when, and where operands can be 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 can be stacked vertically

TYPE= { IN / OUT / INOUT }

or appear on one line:

TYPE={IN|OUT|INOUT}

Both expressions are equivalent. Note how the vertical bar is used to separate alternative values that appear on one line.

An underscored value means that if no value for that operand is selected, 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:

TYPE={IN|<u>OUT</u>|INOUT}

Here, OUT is the assumed value. If the TYPE operand is omitted, TYPE=OUT will be assumed by the macro expansion program.

Brackets ([ ]) denote optional operands. In the following example, the FORM operand is optional.

TYPE={IN|<u>OUT</u>|INOUT}
[,FORM=LG]

An ellipsis (...) indicates that the operand value that precedes it can be repeated any number of times. An operand appearing as

STATION=terminal name,...

in the operands column could, for example, be coded as

STATION=BOS,NYC,PHLY,CLRN,PIT,CGO

Parentheses, equals signs, and uppercase character strings 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 non-blank character, but it cannot be a character of an operand. Comments must by separated for operands by at least one blank. Comments can be intermingled with the operands.

```
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 descriptions may also include:

A description of the format in which the operand should be coded. This description is not provided when the operand's format is apparant from its function or the way it is shown in the assembler format table. For example, the format of such operands as

RPL=rpl address
or
COPIES=quantity

is not described, since the programmer can assume that any assembler expression that will be resolved to an address, including register notation, can be coded with the RPL operand; or that any numerical value (or expression that will be equated to a numerical value) can be coded with the COPIES operand.

A format description *is* provided when restrictions exists that might not be obvious – restrictions such as "register notation cannot be used" or "code the name as a character constant." If any of the terms used in the format descriptions are unclear, refer to *OS/VS and DOS/VS Assembler Language.*

An example showing how the operand is coded or used. 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.

Notes concerning restrictions on how the operand is used, or information that relates indirectly to the operand.

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 may be far simpler when it is actually coded.

Return of Status Information

Most of the macro instructions post return codes in registers and 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.

The Macro Instruction Descriptions

This section contains detailed descriptions of all the macro instructions that are part of the VTAM language. The descriptions are arranged alphabetically.

There are two programming considerations that are not included in the macro instruction descriptions, because they apply to all of the macro instructions that are executable: First, register 13 must always contain the address of an 18-word save area when the macro instruction is executed. Secondly, registers 0, 1, 14, and 15 are used by the macro instruction.

ACB -- Create an Access Method Control Block

The ACB macro instruction builds an ACB control block. The ACB indentifies the application program to VTAM and the teleprocessing network.

Every application program must be defined by the installation before the program can use VTAM to interact 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 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.

The ACB can also provide two indicators that govern the processing that VTAM subsequently performs when requests are made of it:

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 macro instruction.

The application program can ask VTAM to queue logon requests. VTAM will then queue any logon requests that are made after the ACB is opened.

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.

Issuing an ACB macro instruction causes an ACB to be built during program assembly. The ACB can also be built during program execution with the GENCB macro instruction. See the GENCB macro for a description of this facility.

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

symbol

Function: Provides a name for the macro instruction. This name can be used in the ACB operand of an RPL macro instruction.

AM=VTAM

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

APPLID=address of application program's symbolic name

Function: Links the ACB 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.

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

Note: The area pointed to by this operand must begin with a one- byte length indicator, followed by an eight-byte field containing the EBCDIC application program name. The name must be left-justified and padded to the right with blanks. The length indicator must be set to eight.

PASSWD=password address

Function: 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.)

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

Note: The area pointed to by this operand must begin with a one-byte length indicator, followed by an eight-byte field containing the EBCDIC password. The password must be left-justified and padded to the right with blanks. The length indicator must be set to eight.

EXLST=exit list address

Function: 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 macro instruction. See that macro for descriptions of these events.

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

Note: More than one ACB can indicate the same exit list.

MACRF={ LOGON | NLOGON }

Function: Indicates whether or not the application program wants logon requests to be queued for it. MACRF=LOGON causes VTAM to queue logon requests for the application program as they occur after the ACB is opened. MACRF=NLOGON indicates that no queuing of logon requests is to occur.

Note: A logon request 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 request connection between the application program and the terminal. A queued logon request 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 an active LOGON exit list routine (see EXLST macro), that routine is entered whenever a logon request is queued. This routine can issue the OPNDST macro to request connection with the terminal and satisfy the logon request.

Example

```
ACB1     ACB     AM=VTAM,                                  X
                 APPLID=NAMEPASSWD=PASFLD,                 X
                 EXLST=EXLST1,MACRF=LOGON
                 .
                 .
                 .
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 list routine indicated in EXLST1.

CHANGE -- Change a NIB's PROC Option or USERFLD Data

This macro instruction causes modifications to the PROC and USERFLD fields of a NIB to become effective.

When an OPNDST macro instruction is executed, the contents of these NIB fields are moved into internal VTAM control areas. If the application program later wants to change the fields in effect for a NIB, altering the NIB to reflect these changes will not suffice. 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 and/or USERFLD fields have been changed, and whose MODE field has been set to BASIC. 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    NIB=NIB4,USERFLD=NYC,MODE=BASIC                X
         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 keyword=new value]...**

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the RPL whose NIB field contains the address of the NIB that has been modified.

rpl keyword=new value

Function: Indicates an RPL field to be modified, and the *new value* that is to be contained within it.

Format: For *rpl keyword* 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.

Note: See the RPL macro instruction for a list and explanation of all the RPL fields that affect CHANGE, and thus might be modified here.

Return of Status Information

After the CHANGE operation is completed, the contents of register 15 indicate one of the following hexadecimal values:

0 If the ASY OPTION code is in effect, VTAM has accepted the CHANGE request. If the SYN option code is in effect, CHANGE processing has been completed successfully.

4 The CHANGE request cannot be accepted because the RPL is currently in use by another request, or the terminal is not connected to your application program. The RPL's FDBK field has not been set to indicate this error. If an active LERAD exit list routine exists, it has been invoked. See The LERAD description in the EXLST macro instruction.

8 A logical error occurred; the FDBK field can be examined to determine the nature of this error. If an active LERAD exit list routine is available, it has been invoked. (This return code is possible only when the SYN option code is in effect; if the request is asynchronous, this return code results when CHECK is issued.)

C A physical error occurred; the FDBK field can be examined to determine which one it was. (Physical errors are possible with CHANGE because VTAM engages in I/O activity with the 3704 or 3705 communications controller during CHANGE processing.)

1C VTAM cancelled the operation; the second byte of the FDBK field is set indicating the reason.

CHECK -- Check Request Status

When connection, simulated logon, or I/O is requested and asynchronous handling has been indicated in the associated RPL, VTAM schedules the request and returns control to the application program. When the requested operation is completed, VTAM must somehow notify the application program of that fact.

If the request's RPL contains an RPL exit routine address (see the RPL macro description), VTAM automatically invokes this routine when the request is completed. Alternatively, the application program can indicate an ECB work area (see the ECB operand in the RPL macro instruction description) that VTAM posts when the request is completed. The only way the application program can then determine when completion occurs is to check this ECB to verify that it has been posted. The CHECK macro instruction provides this facility.

CHECK causes program execution to wait until the operation is completed. If the operation just completed involved a logical or hardware error, and active LERAD or SYNAD exit routines exist, VTAM invokes one of these routines before returning control to the application program. CHECK also sets the RPL to an inactive (I/O-complete) status so that the RPL can be revised.

CHECK should be used in an RPL exit routine, but not to synchronize the application program with the completion of the request. (The fact that the routine has been invoked is itself evidence that the request has been completed.) What CHECK can still accomplish in this situation however, is the automatic invocation of LERAD or SYNAD exit routines when the request is completed with a logical or hardware error, and the setting of the RPL to an inactive status.

Another important use of CHECK (for either the ECB or the RPL exit routine methods of asynchronous request handling) is to supply a return code that indicates how the requested operation completed. This return code is identical to that returned in the first byte of the RPL's feedback field. The possible return codes are listed at the end of this macro instruction description.

The use and function of the CHECK macro instruction can thus be considered according to its location – either within an EXIT routine or outside of it:

A CHECK issued *outside* of an RPL exit routine checks the ECB indicated in the RPL. The RPL must not indicate an RPL exit routine address. CHECK examines the ECB, posts a return code in register 15, and if appropriate automatically invokes LERAD or SYNAD exit routines.

A CHECK issued *within* an RPL exit posts a return code, and, if appropriate, invokes the LERAD or SNAD exit list routine.

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

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

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

If CHECK is being issued within an RPL exit routine, the RPL's EXIT field must contain the address of the RPL exit routine. If CHECK is not being issued within an RPL exit routine, the RPL's ECB field must point to a fullword in storage to be used as an ECB (or, if neither the ECB nor the EXIT operand has ever been used for the RPL, the ECB field itself is used as an ECB).

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 I/O error, the LERAD or SYNAD exit list 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 I/O error, CHK1 causes the LERAD or SYNAD exit routine to be invoked.

Return of Status Information

When CHECK processing has been completed, register 15 contains one of the following hexadecimal values:

- 0 The request for which this CHECK was issued has been completed successfully.
- 4 The CHECK macro was issued outside of an RPL exit routine, yet the RPL used by the CHECK macro indicated an RPL exit routine address. The RPL for CHECK issued outside of an RPL exit routine must indicate the location of an ECB and not the location of an RPL exit routine. Since this is a logical error, the LERAD exit list routine (if an active one exists) has been invoked.
- 8 A logical error occurred; check the FDBK field of the RPL to determine the nature of this error. This error applies to the operation being checked, not to the CHECK request itself (had there been an error with the latter, a return code of 4 would have been returned instead).
- C A physical error occurred; the FDBK field of the RPL will reveal the specific nature of the problem. This error applies to the operation being checked, and not to the CHECK request itself. The SYNAD exit list routine (if an active one exists) has been invoked.
- 10 A conditional request has been completed with no action taken, since the condition was not met.
- 14 A special condition exists -- for example, an RVI line control character has been received, or an error lock has been set for the device. The second byte of the FDBK field can be examined to determine the specific condition.
- 18 The requested operation was canceled by a RESET request.
- 1C VTAM canceled the operation; the second byte of the FDBK field is set indicating the reason.

CLOSE -- Close One or More ACBs

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

VTAM no longer accepts any requests of any kind 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. Thus logon requests can no longer be directed towards and queued for the ACB. Insofar as terminals requesting logon are concerned, the portion of the application program represented by the ACB ceases to exist when CLOSE is issued.

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.

The CLOSE macro instruction can be applied to more than one ACB.

Name	*Operation*	*Operands*
[symbol]	**CLOSE**	**acb address[, acb address]...**

symbol

Function: Provides a name for the macro instruction.

acb address

Function: Indicates the ACB that is to be disconnected from VTAM.

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/VS1 and OS/VS2 users can also include DCBs.

Example

```
CLOSE123    CLOSE  ACB1,ACB2,(7)
```

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 contains a code indicating whether or not the CLOSE processing has completed successfully. Successful completion (meaning that all ACBs specified in the macro instruction have been disconnected from VTAM) is indicated by a return code of 0. Unsuccessful completion is indicated by any return code other than 0.

If unsuccessful completion is indicated, the application program must examine a field in the ACB to determine the nature of the error encountered by CLOSE. If the CLOSE macro specified more than one ACB, all ACBs must be checked.

The ACB field to be checked is a four-byte area called the ERROR field. It is not a field that the application program should modify -- 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 macro instruction. For example:

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

The hexadecimal value in the ERROR field indicates the nature of the error encountered during CLOSE processing:

- 00 CLOSE successfully closed this ACB.
- 04 The ACB does not belong to the job step that issued the CLOSE macro instruction.
- 08 A CLOSE macro instruction has already been successfully issued for this ACB.

CLSDST -- Disconnect a Terminal from the Application Program

The CLSDST (close destination) macro instruction constitutes a request for VTAM to break a connection that exists between the application program and a specified terminal.

The terminal to be disconnected is specified either with 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 represented by that NIB 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.)

If the terminal has been defined by the installation as a dial-out terminal (by specifying CALL=OUT in the LINE or GROUP macros during VTAM definition), CLSDST causes a dial-line disconnection to occur. There is one exception, however; if an automatic logon request was indicated for the terminal by the installation, the terminal is not disconnected.

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

The CLSDST macro instruction can optionally be used to request that VTAM reconnect a terminal to another application program in addition to disconnecting it. This option is implemented by setting the PASS option code in CLSDST's RPL. If this option is used, VTAM first disconnects the terminal and then generates a logon request for it. Your application program must indicate which application program is to receive the logon request. A logon message from a data area in your program can also be sent with the logon request. (The data area containing the logon message can be reused as soon as CLSDST has been completed.)

If a logon request 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 eight-byte field, left justified, and padded to the right with blanks. If a logon message is also to be sent with the logon request, the AREA and RECLEN fields must indicate the location and length of the message. If a message is not to be sent, the AREA and the RECLEN fields must be set to 0.

(For programs running under OS/VS1 or OS/VS2, the use of CLSDST with PASS must be authorized by the installation. When CLSDST with PASS is issued, VTAM either checks whether the installation has indicated authorization or invokes an installation-written routine that determines if the request is authorized.)

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 installation indicated during VTAM definition that automatic logon requests are to be generated, VTAM connects 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, CLSDST is not the appropriate macro instruction to use. The CLOSE macro instruction should be used, since it disconnects all of the terminals connected via a given ACB. CLOSE, however, cannot be used to generate logon request on behalf of a terminal when the terminal is disconnected.

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

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL to be used during CLSDST processing. Either the ARG field of this RPL must be filled with a terminal's CID, or the NIB field must be set to point to the NIB whose associated terminal is to be disconnected.

rpl keyword=new value

Function: Indicates an RPL field to be modified, and the new value that is to be contained within it.

Format: For *rpl keyword* 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.

Note: All of the RPL fields that have a unique effect on CLSDST (and thus might likely be modified here) are discussed above. Check Figure 6 in the RPL macro instruction description for a list of all of the RPL fields that apply for the CLSDST macro instruction.

Examples

```
CL1  CLSDST  RPL=RPL1,          RPL1 MODIFIERS FOLLOW:                    X
             ACB=ACB1,                                                    X
             NIB=NIB3,            (TERMINAL TO BE DISCONNECTED)           X
             AAREA=APPLNAME,      (APPLICATION TO RECEIVE LOGON REQUESTS) X
             AREA=LGNMSG,RECLEN=60,     (LOGON MESSAGE)                   X
             ECB=POSTIT1,OPTCD=(ASY,PASS)
```

CL1 disconnects the terminal represented in NIB3, and generates a simulated logon request for it; the logon request is directed at the application program whose symbolic name (APPL entry name as defined by the installation) has been placed in APPLNAME. This macro instruction also sends a 60-byte logon message from LGNMSG with the logon request.

```
CL2  CLSDST  RPL=RPL2,          RPL2 MODIFIERS FOLLOW:                    X
             ARG=(3),             (TERMINAL TO BE DISCONNECTED)           X
             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 any logon message.

```
CL3  CLSDST  RPL=RPL3,          RPL3 MODIFIERS FOLLOW:                    X
             NIB=NIB6             (TERMINAL TO BE DISCONNECTED)           X
             AAREA=APPLNAME,      (APPLICATION TO RECEIVE LOGON MESSAGE)  X
             AREA=0,RECLEN=0,       (NO LOGON MESSAGE)                    X
             ECB=POSTIT3,OPTCD=(ASY,PASS)
```

CL3 disconnects the terminal represented by NIB6 and generates a simulated logon request for it that is directed at the application program represented by APPLNAME. Since the AREA and RECLEN fields are being set to 0, no logon message is sent.

Return of Status Information

After the CLSDST operation is completed, register 15 contains one of these hexadecimal values:

0 If the ASY option code is in effect, VTAM has accepted the CLSDST request. If the SYN option code is in effect, CLSDST processing was completed, and all indicated terminals were successfully disconnected.

4 The request cannot be accepted because the RPL is currently in use by another request; or the terminal being disconnected is not connected to your application program. The RPL's RDBK field has not been set. If an active LERAD exit list routine exists, it has been invoked.

8 A logical error (see Appendix A) occurred; the FDBK field can be examined to determine which kind it was. If an active LERAD exit list routine exists, it has been invoked. This return code is possible only when the SYN option code is in effect; otherwise, this code is returned when a CHECK macro instruction is issued.

1C VTAM canceled the operation; the second byte of the FDBK field is set indicating the reason.

DO -- Initiate LDO-specified I/O Operations

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 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 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. The operations available via DO are these:

Copy the contents of a 3270 display unit buffer into the buffer of any printer or display unit attached through the same control unit (implemented with the COPYLBM and COPYLBT LDOs).

Read the entire contents of a 3270 display unit buffer (implemented with the READBUF LDO).

Send a positive response with leading graphic characters to a 3735 terminal or to a System/3 or System/370 CPU and then read the device's next block of data; or send a negative response with leading graphic characters to one of these devices and then re-read the block of data (implemented with the WRTPRLG, WRTNRLG, and READ LDOs).

Write data beginning with a block of heading characters to a 3735 terminal or to a System/3 or System/370 CPU (implemented with the WRTHDR LDO).

These I/O operations are more fully explained in the LDO macro instruction description.

If DO is processing a list of LDOs, VTAM continually updates the AAREA field of the RPL to indicate the address of the LDO currently being used. Should an error occur during DO processing, this updating ceases, and AAREA indicates the LDO involved with the error. Register 15 is set to indicate that DO did not complete its function successfully.

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

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL whose AREA field contains the address of an LDO or group 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 keyword=new value

Function: Indicates a field of the RPL to be modified and the new value that is to be contained within it.

Format: For *rpl keyword* 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.

Note: See Figure 5 in the RPL macro instruction description for a list of the RPL fields that are applicable for the DO macro instruction.

Example

```
DOR2LDO   DO    RPL=RPL1,          RPL1 MODIFIERS FOLLOW:    X
                AREA=(2),ARG=(3),                            X
                EXIT=DONE,OPTCD=(SPEC,ASY)
```

DOR2LDO initiates whatever operations are indicated by the LDO (or list of LDOs) currently 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 DOR2LDO before the operation is actually performed.

Return of Status Information

Once DO processing is finished, these sources of status information can be checked.

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

The USER field of the RPL: When a NIB is established, 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 had been placed in USERFLD by the user is placed in the USER field of the RPL by VTAM.

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

The FDBK field of the RPL: Unless a hexadecimal value of 4 was returned in register 15, the FDBK (feedback) field describes the completion status of the DO processing. See Appendix A for a description of the FDBK field.

Register 15: One of the following hexadecimal values are returned:

- 0 The DO request has either been accepted (ASY option code in effect) or all of the operations performed by DO have been completed successfully (SYN option code in effect).
- 4 The DO request cannot be accepted because the RPL is currently in use by another request, or the CID in the ARG field is not valid (for example, does not represent a terminal currently connected to your application program). The RPL's FDBK field has not been set. If an active LERAD exit routine exists, it has been invoked.
- 8 A logical error (see Appendix A) occurred; the FDBK field can be examined to determine which one it was. If an active LERAD exit list routine exists, it has been invoked. (This code can be returned only when the SYN option code is in effect.)
- C A physical error (see Appendix A) occurred; the FDBK field can be examined to determine which one it was. If an active SYNAD exit list routine is available, it has been invoked. (This code will be returned only when the SYN option code is in effect.)
- 1C VTAM canceled the operation (or one of the operations) indicated by the LDO (or LDOs) used by the DO macro instruction. The second byte of the FDBK field is set indicating the reason.

EXLST -- Create an Exit List

The EXLST macro instruction builds a list of routine addresses. Each operand in this macro instruction represents a circumstance in 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 circumstance that it handles occurs. The SYNAD operand supplies the address of a routine that handles physical errors, the ATTN operand supplies the address of an attention-interruption handler, and so forth.

Routines can be marked active or inactive – active meaning that the routine should be invoked, inactive meaning that it should not. Since the exit list cannot be extended during program execution, space must be reserved in the list for entries that may eventually be needed. Marking an exit routine inactive in effect reserves space for that exit in the list. During program execution, the MODCB macro instruction can be used to insert a valid address and mark it active.

All but two of the routines that can be indicated in an exit list are invoked as a result of an event initiated *outside* of the application program. All of these exit routines are invoked asynchronously – that is, at the time the event occurs. 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.

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

For all exit list 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.

The address of the exit list created by the EXLST macro instruction is placed in the EXLST field of an ACB (see the ACB macro instruction for details). More than one ACB can point to the same exit list, as long as the ACBs are all in the same object module. In this situation, however, the routines indicated in the exit list should be reenterable. (Exit list routines should also be reenterable if the LE or GE attribute has been added to the routine's address.) All of the exit list routines must likewise occur in the same object module.

Name	*Operation*	*Operands*
[symbol]	EXLST	AM=VTAM [,LERAD= (address [,{A|N}])] [,SYNAD= (address [,{A|N}])] [,UNSIP= (address [,{A|N}][,{E|LE|GE}])] [,ASYIP= (address [,{A|N}][,{E|LE|GE}])] [,TPEND= (address [,{A|N}][,{E|LE|GE}])] [,RELREQ= (address [,{A|N}][,{E|LE|GE}])] [,LOGON= (address [,{A|N}][,{E|LE|GE}])] [,LOSTERM=(address [,{A|N}][,{E|LE|GE}])] [,ATTN= (address [,{A|N}][,{E|LE|GE}])]

symbol

Function: Provides a name for the macro instruction. This name can be specified as the value of the EXLST operand of the ACB macro instruction.

AM=VTAM

Function: Identifies the exit list generated by this macro instruction as a VTAM exit list (as distinguished from a VSAM exit list). This operand must be coded for application programs running under DOS/VS; for application programs running under OS/VS1 or OS/VS2, this operand is ignored.

The operands of the EXLST macro may be qualified by the following operand values. The assembler format table above indicates which values can be specified for each operand. (The parentheses can be omitted from any EXLST operand that does not have any of these values coded with it.)

A|N
These values indicate whether or not the routine is to be entered.

A
The exit list routine is active. VTAM will schedule the routine.

N
The exit list routine is inactive. VTAM will not schedule the routine, even if the entry contains a valid address.

Active addresses can be made inactive and inactive addresses made active with the MODCB macro instruction. If a SYNAD exit is to be made active, for example, the MODCB macro to activate it could be coded as follows:

```
MOD1    MODCB    AM=VTAM,EXLST=LIST1,SYNAD=(PGM4,A)
                          or
MOD2    MODCB    AM=VTAM,EXLST=LIST1,SYNAD=(,A)
```

(MOD1 changes the exit routine address and marks it active; MOD2 merely marks an existing address active.)

E|LE|GE
These values indicate whether or not the exit list routine is to be executed in supervisor state with a high dispatching priority.

Note: Exit list routines can be executed in supervisor state only if the program is running under OS/VS2 and only if the application program has been authorized to do so by the installation.

E
The exit list routine is not to be executed in supervisor state. This is the assumed attribute for an exit list address, and is the *only* one of these three attributes that can be used by programs running under DOS/VS or OS/VS1.

LE OS/VS2 Only
The exit list routine is scheduled for execution under a *local SRB* (system request block). All READ, WRITE, and SOLICIT requests must have their RPL's BRANCH field set to YES. This type of scheduling should be used only by those who are thoroughly familiar with the restrictions that apply to routines scheduled under an SRB.

The use of LE must be authorized for the application program by the installation. If an unauthorized program attempts to open an ACB that points to an exit list containing an address with an LE attribute, that ACB will not be opened. An additional authorization check is made when the exit list routine is scheduled.

GE OS/VS2 Only
The exit list routine is scheduled for execution under a *global* SRB. The effect of GE is otherwise exactly the same as for LE, including the requirement for installation authorization. This type of scheduling should also be used only by those who are thoroughly familiar with the restrictions that apply to routines scheduled under an SRB.

LERAD= (address [,{A|N}])

Function: 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 a request is made that is inherently contradictory – like attempting to read data from an output- only device. (Errors that occur because of hardware malfunctions are not logical errors; they are handled by the SYNAD exit list routine.)

If the SYN option code is in effect when the error occurs, the LERAD routine is entered immediately; but if the ASY option code is in effect, the routine will not be scheduled until a CHECK macro instruction is issued for the operation in which the error occured. One exception: If the ASY option code is in effect and a request fails because its RPL is currently in use, the LERAD exit routine is scheduled immediately.

If the application program has no active LERAD exit list routine and a logical error occurs, VTAM simply returns control to the next sequential instruction. VTAM places a return code in register 15 (indicating a logical error) regardless of whether a LERAD routine is invoked before control is finally returned.

When the LERAD exit list routine returns control to VTAM, VTAM leaves register 0 intact so that the routine can pass information back in this register to the main part of the application program.

Note: When the LERAD routine receives control, the general purpose registers contain the following:

Register 1 – the address of the RPL associated with the request. If the high-order bit of this address is off, the RPL's FDBK field indicates the type of logical error that occured. If the high-order bit is on, VTAM was unable to place an indicator in the FDBK field specifying the reason for the error. This happens in two cases: Either a macro has been issued where RPL is already in use, or CHECK has been issued for a request whose RPL exit routine has not yet been scheduled.

Register 14 – the address in VTAM to which the LERAD exit list routine can branch when it is through processing. When the exit list routine branches to this address, VTAM handles the returning of control to the next sequential instruction in the application program following the request. (The LERAD routine can itself return control directly to the next instruction by first restoring the registers from the save area pointed to by register 13 and then branching on register 14.)

Register 15 – the address of the LERAD routine.

The contents of register 0 may be modified by VTAM before the LERAD exit routine is invoked. Registers 2-13 are not modified by VTAM.

SYNAD= (address [,{A|N}])

Function: Indicates the address of a routine that is entered if an unrecoverable input or output error occurs during an I/O operation. This is called a *physical error.* (Errors that result from invalid requests are handled by the LERAD exit routine.)

If the SYN option code is in effect when the error occurs, the SYNAD exit routine is entered immediately; 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.

Before the SYNAD exit routine is given control, VTAM fills in the FDBK (feedback) field of the RPL. This field indicates the nature of the I/O error that caused the routine to be invoked. See Appendix A for a description of the FDBK field.

The SYNAD exit routine can analyze the FDBK field and attempt to fix the error. For example, if FDBK indicates that the terminal's error lock is set, the SYNAD exit routine can issue a RESET macro instruction to reset the lock. Or, if FDBK indicates that the terminal is no longer connected, the SYNAD exit routine can strike that terminal from a list of those with which it is communicating.

If the application program has no active SYNAD exit list routine and a physical error occurs, VTAM simply returns control to the next sequential instruction. VTAM returns a code in register 15 (indicating a physical error) regardless of whether a SYNAD routine is invoked before control is finally returned.

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

Note: When the SYNAD routine receives control, the general purpose registers contain the following:

Register 1 -- the address of the RPL associated with the I/O request.

Register 14 – the address in VTAM to which the SYNAD exit list routine can branch when it is through processing. When the exit routine branches to this address, VTAM handles the return of control to the next sequential instruction following the I/O request (or following the CHECK macro issued for the I/O request). The SYNAD exit list routine can return control directly to the next instruction by first restoring the registers from the save area pointed to by register 13 and then branching on register 14. LERAD and SYNAD are the only exit list routines that do not have to branch on the address contained in register 14.

Register 15 – the address of the SYNAD routine.

The contents of register 0 may be modified by VTAM before the SYNAD exit routine is invoked. Registers 2-13 are not modified by VTAM.

UNSIP= (address [,{A|N}][,{E|LE|GE}])

Function: Indicates the address of a routine that is entered when unsolicited input is received from a terminal.

Unsolicited input is input received from a terminal connected to an application that did not issue a READ or SOLICIT macro for that terminal.

Unsolicited input occurs when a device connected to the application program responds to a solicit operation that the application program did not direct to it. This may result when a read or solicit request directed to one component of a terminal system (like the 3270 Information Display System) causes the control unit to solicit data from all of its attached input components. If the component that was inadvertently solicited responds, the UNSIP exit routine of the application program to which it is connected is invoked. If there is no UNSIP exit list routine, the data is lost.

Note: The application program must issue a read request if it wants to obtain the unsolicited input sent to it; until it does so, the input remains in VTAM buffers. Before the UNSIP routine can issue the read request, it must first build an RPL for it. When the UNSIP routine receives control, register 1 contains the address of a parameter list that contains the following information needed for the RPL:

The first word of the parameter list contains the address of an ACB. This ACB is the ACB of the application program to which the unsolicited input was directed. This ACB should be pointed to by the ACB field of the RPL being built.

The second word contains the CID of the terminal that sent the unsolicited input. This CID should be placed in the ARG field of the RPL being built.

The third word of the parameter list contains whatever has been placed in the USERFLD field of the NIB used when the terminal was connected.

The fourth word contains the number of bytes of unsolicited data.

When the UNSIP exit routine receives control, the other general purpose registers contain the following:

Register 14 – the address in VTAM to which the UNSIP routine must branch when it is through processing. VTAM handles the return of control to the instruction that was about to be executed when the interruption for unsolicited input occurred.

Register 15 – the address of the UNSIP routine.

The contents of the remaining registers (0 and 2-13) may be modified by VTAM before the UNSIP exit routine is invoked.

ASYIP= **(address [,{A|N}][,{E|LE|GE}])**

Function: Indicates the address of a routine to be entered when a block of data arrives in VTAM buffers as a result of a SOLICIT macro instruction for which the CA option code is in effect. The ASYIPX processing option must be in effect for this exit routine to be invoked.

The ASYIP exit routine provides a way for VTAM to notify the application program that solicited data has arrived. (Since solicit requests are considered complete as soon as they are scheduled, and not when data arrives, the application program cannot use the usual ECB posting or RPL exit routine invocation as a way of finding out when the data arrives.)

The processing to be performed in the ASYIP exit list routine is at the discretion of the user. The exit routine may issue a READ macro instruction having a SPEC

option code in effect to retrieve the data. In any event, the data remains in VTAM buffers until it is retrieved.

Note: When the ASYIP routine receives control, register 1 contains the address of a four-word parameter list:

The first word of the parameter list contains the address of an ACB. This ACB is the ACB of the application program to which the input data was sent. The ACB field of any RPL that will be used for subsequent I/O requests for this terminal should contain this address.

The second word contains the CID of the terminal that sent the input data. The ARG field of any RPL built for subsequent I/O with the terminal must contain this CID.

The third word contains whatever has been placed in the USERFLD of the NIB associated with that terminal.

The fourth word contains the number of bytes of data received by VTAM.

The other general purpose registers contain the following:

Register 14 – the address in VTAM to which the ASYIP routine must branch when it is through processing. VTAM will handle the return of control to the instruction following the instruction that was about to be executed when the ASTIP interruption occurred. Register 15 – the address of the ASYIP routine.

Register 15 – the address of the ASYIP routine.

The contents of the remaining registers (0 and 2-13) may be modified by VTAM before the ASYIP exit routine is invoked.

TPEND= **(address [,{A|N}][,{E|LE|GE}])**

Function: Indicates the address of a routine to be entered when the network operator issues a HALT command, or when VTAM itself shuts down. If the operator issues a HALT command to cause a flush closedown, any pending I/O for the application program is allowed to be completed. The TPEND exit routine can, for example, interrupt any pending solicit operations and write messages to all connected terminals, informing them that they are about to be disconnected.

If the operator issued a HALT command to cause a quick closedown, or VTAM itself shuts down, any pending I/O operations are canceled, and the appropriate RPL FDBK fields are posted to indicate the reason for their premature completion. In this situation, the TPEND exit routine cannot send or receive any data from the connected terminals, and can only disconnect them all with the CLOSE macro instruction.

Note: When the TPEND exit routine receives control, register 1 contains the address of a two-word parameter list:

The first word of the parameter list contains the address of an ACB. This ACB is the ACB of the application program being shut down.

The hexadecimal value contained in the second word indicates the reason for the shutdown:

0 The operator issued a HALT command, causing an ordinary closedown.

4 The operator issued a HALT command, causing a *quick* closedown.

8 VTAM is shutting down.

The other general purpose registers contain the following:

Register 14 – the address in VTAM to which the TPEND routine should branch when it is through processing.

Register 15 – the address of the TPEND routine.

The contents of the remaining registers (0 and 2-13) may be modified by VTAM before the TPEND exit routine is invoked.

RELREQ= (address [,{A|N}][,{E|LE|GE}])

Function: Indicates the address of a routine that is entered when another application program requests connection to a terminal that is currently connected to your application program. This occurs when the other application program issues an OPNDST macro instruction with ACQUIRE and RELRQ processing options in effect, or when the other application program issues a SIMLOGON macro instruction 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, 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 issued.

The application program that causes your RELREQ exit list routine to be invoked may have had the CONANY option code in effect for its OPNDST or 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 ignored by the other application program when its request is satisfied by some other terminal. To prevent this problem, follow the CLSDST with an OPNDST or SIMLOGON request of your own, with the NRELRQ processing option in effect. Then if the other application program is ignoring your just-released terminal, you get it back.

Note: When the RELREQ exit routine receives control, register 1 contains the address of a three-word parameter list:

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 CID of the terminal.

The third word contains a pointer to the symbolic name of the requesting application program.

The other registers contain the following:

Register 14 – the address in VTAM to which the RELREQ routine should branch when it is through processing. VTAM will handle the return of control 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) may be modified by VTAM before the RELREQ exit routine is invoked.

LOGON= **(address [,{A|N}][,{E|LE|GE}])**

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

VTAM will queue a logon request (1) when a terminal issues a logon request and the application program to which the terminal is currently connected issues a CLSDST macro instruction with OPTCD=PASS, (2) when an appication program issues a SIMLOGON macro instruction on behalf of the terminal, or (3) when an application program opens its ACB, if the installation indicated during VTAM definition that logon requests should automatically be generated on behalf of specified terminals.

If a terminal has been defined by the installation as a dial-in terminal (CALL=IN specified in the LINE or GROUP definition macro), the LOGON exit list routine is scheduled when the terminal operator dials in. If a terminal has been defined by the installation as a dial-out terminal (CALL=OUT), the LOGON exit list routine is scheduled when the application program opens its ACB. (The terminal will not be dialed, however, until the terminal is connected and the first I/O request is directed to it.)

Regardless of the mechanism by which the LOGON exit list 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.

The LOGON exit routine can issue an INQUIRE macro instruction to obtain the logon message supplied by the terminal making the logon request. If the routine determines from this logon message that connection should be requested, it may wish to establish that connection. This would be accomplished by using information passed to the LOGON exit list routine, along with information obtained with the INQUIRE macro instruction, to build a NIB and an RPL, and by then issuing the OPNDST macro instruction with ACCEPT and SPEC option codes to request the connection.

Note: The LOGON exit list routine is entered only if MACRF=LOGON has been specified for the ACB, and the application program has not closed its logon queue with a SETLOGON (OPTCD=QUIESCE or OPTCD=STOP) macro instruction.

When the LOGON exit list routine receives control, register 1 contains the address of a four-word parameter list:

The first word of the parameter list 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 logon message.

The second word contains the address of the eight-byte symbolic name of the terminal making the logon request. 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 terminal's entry is either a TERMINAL or COMP entry, or, if the terminal is a dial-in terminal without an automatic ID verification feature, a UTERM entry. TERMINAL, COMP, and UTERM are VTAM definition macros used by the installation to build entries in the resource definition table.)

The third word of the parameter list contains the address of the logon message. This address is useful only for programs running under DOS/VS, or programs running under an SRB in OS/VS2; for all other application programs the logon message is fetch-protected, and the INQUIRE macro instruction must be used to obtain the logon message.

The fourth word contains the length of the logon message sent by the terminal. This length should be used with the LENGTH operand of any INQUIRE macro instruction used to obtain the logon message.

The 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) may be modified by VTAM before the LOGON exit routine is invoked.

LOSTERM=(address [,{A|N}][,{E|LE|GE}])

Function: Indicates the address of a routine to be entered when a switched-line disconnection occurs or when the network operator takes the terminal from the application program with a VARY command.

When the LOSTERM exit list routine is entered, the application program can no longer communicate with the terminal. The LOSTERM exit list routine may want to inform the main part of the application program that the terminal has been lost. It should eventually disconnect the terminal with a CLSDST macro instruction. The information needed to do this is available to the exit routine via parameters passed to it when it is given control.

The LOSTERM routine is scheduled only if there are no pending I/O requests for the terminal. If I/O requests are pending, VTAM cancels the I/O operation and posts a return code in the RPL's FDBK field indicating that it has done so. If the application program has no LOSTERM exit list routine, the application program will not discover the disconnection until it issues an I/O request for the terminal.

Note: When the LOSTERM exit list routine receives control, register 1 contains the address of a four-word parameter list.

The first word of the parameter list 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:

0 A switched-line disconnection occurred.

4 The network operator issued a VARY command.

The other general purpose registers contain the following:

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

The contents of the remaining registers (0 and 2-13) may be modified by VTAM before the LOSTERM exit list routine is invoked.

ATTN= **(address [,{A|N}][,{E|LE|GE}])**

Function: Indicates the address of a routine to be entered when a terminal connected to the application program issues an attention interruption and no read or write operation is pending or in progress.

Such an attention interruption causes an error lock to be set for the terminal by the CPU or the 3704 or 3705 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 active ATTN routine to be invoked, the attention interruption is ignored and the error lock is automatically reset.

Note: 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 installation indicated during VTAM definition that the 3704 or 3705 communications controller is to react to attention interruptions.

When the ATTN routine receives control, register 1 will contain the address of a three-word parameter list:

The first word of the parameter list 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 had been placed in the USERFLD of the NIB associated with the terminal.

The other general purpose registers contain the following:

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

Register 15 – the address of the ATTN exit routine.

The contents of the remaining registers (2-13) may be modified by VTAM before the ATTN routine is invoked.

Example

```
EXLST1  EXLST LERAD=LERADPGM,SYNAD=SYNADPGM, SEE NOTE 1  X
              TPEND=(,N),RELREQ=(,N),          SEE NOTE 2  X
              AM=VTAM
```

Note 1: Both the LERAD and SYNAD exits are specified as active. This is the same attribute that the exit addresses would have if A had been specified for them.

Note 2: No TPEND or RELREQ exits are being indicated in the exit list, but space for them is being reserved. Later the MODCB macro can be used to insert an address and change the entry from inactive (N) to active (A).

GENCB – Generate a Control Block

The GENCB macro instruction builds an ACB, EXLST, RPL, or NIB control block. The advantage of using the GENCB macro instruction lies in the fact that the control blocks are generated during program execution. (With the ACB, EXLST, RPL, and NIB macro instructions, the control blocks are built during assembly.) Program reassembly will not be required should control block formats be changed during future releases of VTAM. GENCB not only builds the control block during program execution, it can also build the control block in dynamically allocated storage.

The GENCB user specifies the type of control block to be built and the content of fields within it. The operands used to specify the field contents are exactly the same as those used in the macro instruction that builds the control block. For example, this GENCB macro instruction –

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

builds the same exit list as this EXLST macro:

```
EXLST    SYNAD=SYNADPGM
```

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.

Note: 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. Required control block lengths are indicated below in the LENGTH operand description.

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

Name	*Operation*	*Operands*
[symbol]	GENCB	BLK={ ACB | EXLST | RPL | NIB } [, keyword=value]... [, COPIES={ 1 | quantity }] [, WAREA=work area address] [, LENGTH=work area length] [, MF={ L | (E, parameter list address) }]

symbol

Function: Provides a name for the macro instruction. This name can be used by other macro instructions that refer to the control block being built.

BLK={ ACB | EXLST | RPL | NIB }

Function: Indicates the type of control block to be generated.

Note: This operand is required for standard and list forms, but for the execute form it can not be specified.

keyword=value

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

Format: For *keyword* code any keyword that can be used in either the ACB, EXLST, RPL, or NIB macro instruction corresponding to that shown in 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.

Note: Register notation cannot be used for the list form of GENCB

COPIES={ 1 | quantity }

Function: Indicates the number of control block copies to be generated, including the original.

The copies are identical in form and content. They are placed contiguously in storage, whether that storage is the area indicated by the WAREA operand or is dynamically allocated storage.

The length returned in register 0 will be 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.

Format: Register notation cannot be used for the list form of GENCB.

WAREA=work area address

Function: Indicates the location of the storage area in the application program where the control block is to be built. 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. The address of the generated control block is then be placed in register 1.

Format: Code the location of the work area. For the list form of GENCB, register notation cannot be used.

LENGTH=work area length

Function: 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 required length. If the length is sufficient, VTAM generates the control block or blocks and posts the total length in register 0.

The following amounts of storage are required for each block:

	DOS/*VS*	*OS/VS1*	*OS/VS2*
ACB	52	84	84
NIB	44	44	44
RPL	76	104	104
EXLST	70	70	70

Format: Code a numerical value or an expression that will be equated to a numerical value. Expressions involving registers cannot be used for the list form of GENCB.

MF={ L | (E, parameter list address) }

Function: Indicates that either a list form or an execute form of GENCB is to be used.

MF=L

The list form (L-form) of this macro instruction creates a parameter list for later use by the execute form. Because the L-form macro instruction generates only this parameter list, and no executable code, operand forms like register notation or base-displacement expressions are prohibited. Only A-type address constants or absolute values can be used. The user is responsible for branching around the generated parameter list, which is variable in length.

MF=(E,parameter list address)

The execute form (E-form) of this macro instruction can modify the parameter list generated by its list form, and causes control to be passed to VTAM routines when the E-form is executed. The expansion of the execute form provides the executable instructions required to perform parameter list modification and passing of control. The *parameter list address* should specify the location of the list form of the macro instruction. Register notation can be used.

Note: Although the execute form of GENCB can modify the list form's parameter list, it cannot add to it. Therefore if an operand value will not be known until program execution and is to be supplied with the execute form, the list form must also specify that operand and supply a dummy value for it.

For example: Assume than an RPL is to be generated, and the value for the RECLEN operand will become available in register 3 during program execution. To place the contents of register 3 into this RPL field, code a list form like this:

```
LGEN     GENCB   BLK=RPL,RECLEN=0,MF=L
```

and code an execute form like this:

```
EGEN     GENCB   RECLEN=(3),MF=(E,LGEN)
```

Examples

```
GEN1     GENCB   AM=VTAM,BLK=ACB,                    X
                 APPLID=(3 ),EXLST=(6),              X
                 WAREA=ACB1AREA,LENGTH=76
```

GEN1 builds an ACB in ACB1AREA. When GEN1 is executed, register 3 must contain the address of an application program's symbolic name (the name of that application program's entry in the resource definition table), and register 6 must contain the address of the exit list to be pointed to by the ACB.

```
GEN2        GENCB    BLK=RPL,COPIES=10,AM=VTAM
```

GEN2 creates ten RPLs in dynamically allocated storage. The address of the beginning of these RPLs is returned in register 1. 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 next sequential instruction, VTAM indicates the following in these general purpose registers:

Register 0: If the control block was built in dynamically allocated storage, this register contains the length (in bytes) of the control block. If several copies have been built, the length represents the total length. If the control blocks were to have been built in WAREA but the length (LENGTH) was too small, this register contains the correct length needed to build the control block or blocks.

Register 1: This register contains the address of the first byte of the generated control blocks if they were built in dynamically allocated storage.

Register 15: The hexadecimal value in this register indicates completion status:

0 The control blocks were successfully built.

4 Either the statically reserved work area was too small (see register 0 for the required length) or not enough dynamically allocated storage could be obtained.

8 The execute form was used incorrectly. For example, a control block field was being set, but the list form did not indicate that field and supply a dummy value for it.

INQUIRE -- Obtain Terminal Information or Application Program Status

As with many other VTAM macro instructions, various fields in an RPL determine what will happen when INQUIRE is executed. The user coding the INQUIRE macro instruction must indicate a particular RPL and can optionally reset any of its fields.

There are five variations to INQUIRE; an RPL option code (in its OPTCD field) determines which one is used:

OPTCD=LOGONMSG
INQUIRE obtains a logon message from a terminal that has requested logon to the program. This type of INQUIRE is useful in LOGON exit list routines that must evaluate the logon request to determine whether or not connection is to be established with the terminal. (*Note:* A logon message cannot be obtained after OPNDST is issued, and it can only be obtained once.

The RPL's ACB field must indicate the ACB to which the logon request is directed. The NIB field must point to a NIB whose NAME field contains the symbolic name of the terminal issuing the logon request. 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, NIB, and AREALEN fields is passed to the LOGON exit list 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.

OPTCD=DEVCHAR
INQUIRE obtains the device characteristics of a terminal, as they are defined in the resource definition table at the time INQUIRE is executed. These device characteristics can be used to define which processing options the program wants to be in effect for the NIB associated with that terminal. This type of INQUIRE is also appropriate for use in LOGON exit list routines where the program is establishing connection with terminals whose identities are not known during program assembly.

The RPL must indicate the terminal in one of two ways: either the RPL's NIB field must indicate a NIB that contains the symbolic name of the terminal, or the RPL's ARG field must contain the CID of the terminal. These device characteristics are placed in an eight-byte program storage area whose location is indicated in the AREA field. The AREALEN field must be set to eight.

The bits that are set in this field indicate whether the device is an input, output, or input/output device. The specific device type (for example, 3270 display unit) is also indicated.

OPTCD=TERMS
For a given TERMINAL LINE, or GROUP entry in the resource definition table, INQUIRE builds a NIB or list of NIBs in the application program.

The reason for this type of INQUIRE is this: During VTAM definition, the installation can define a line or GROUP entry and associate a set of terminals with that entry. If the application programs builds one NIB that indicates this LINE or GROUP entry in its NAME field, it can then issue INQUIRE to generate NIBs for

all of the LINE or GROUP entry's associated terminals. 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 installation, via the network operator, to vary the set of terminals after the application program has been assembled.

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. This entry must be either a terminal entry (as defined by a TERMINAL macro instruction during VTAM definition) or a group entry (as defined by a GROUP macro) that represents several terminals. 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.

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; the required length will be returned in the RECLEN field. Obtain the storage, then reissue INQUIRE, this time with AREALEN set to the value returned in RECLEN.

Each NIB contains the symbolic name of the terminal, processing options set according to the underscored ('default') values shown for the PROC operand in the NIB macro instruction, and 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, the NIBs are ready to be used for connection.

OPTCD=COUNTS
INQUIRE provides the number of terminals that are currently connected via a given ACB and the number of terminals that have requested logon via that ACB but have not yet been connected.

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

OPTCD=APPSTAT
INQUIRE checks a given application program and set a value indicating:

0 the application program is active and available – that is, represented by an ACB that (1) has been opened, and (2) indicates that logon requests are to be queued, or

4 the application program is inactive (future availability unknown) – that is, not represented by any opened ACB.

8 the application program is active but unavailable – that is, represented by an ACB that has been opened but indicates that logon requests are not to be queued.

(An application program indicates that logon requests are not to be queued by either specifying MACRF=NLOGON in its ACB or issuing a SETLOGON macro instruction.)

The value is returned in a fullword storage area provided by the application program; the address and length of this area must be placed in the AREA and AREALEN fields of the RPL.

The RPL's NIB field must point to a NIB whose NAME field contains the symbolic name of the application program to be checked. (Although the NIB is generally used as a *terminal* control block, note that here it is being used as an application program control block.) The symbolic name 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.)

OPTCD=CIDXLATE
Given a terminal's CID, INQUIRE provides the symbolic name of that terminal.

When a terminal is connected to an application program, the symbolic name of that terminal is converted into a four-byte equivalent called the CID. This CID is inserted into the ARG field of the RPL used for connection, and must be used for all subsequent READ, WRITE, SOLICIT and RESET requests for that terminal.

The CIDXLATE option provides a means of reconverting that CID back into its equivalent symbolic name. The RPL's ARG field must contain the CID when INQUIRE is executed. The symbolic name is returned in the data area indicated by the RPL's AREA field.

OPTCD=TOPLOGON
When a terminal directs a logon request at an application program (ACB), or when a logon request is made on its behalf, the application program may or may not immediately satisfy the request. While the request remains unsatisfied, it is said to be *queued* on the ACB. If unsatisfied requests accumulate, more than one terminal will be queued on that ACB.

The TOPLOGON option provides the symbolic name of the terminal that is currently at the head of this queue. The ACB field of INQUIRE's RPL must indicate the ACB whose logon request queue is to be used. The symbolic name is returned in the data area indicated in the RPL's AREA field.

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

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL that indicates which kind of processing INQUIRE is to perform.

rpl keyword=new value

Function: Indicates an RPL field to be modified, and the new value that is to be contained or represented within it.

Format: For *rpl keyword* 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.

Note: The seven RPL option codes that have a unique effect on INQUIRE (and thus might likely be modified here) are discussed above; check Figure 5 in the RPL macro instruction description for a list of all RPL option codes and fields that apply to the INQUIRE macro instruction.

Examples

```
INQ1     INQUIRE   RPL=RPL1,      RPL MODIFIERS FOLLOW   X
                   OPTCD=APPSTAT,                        X
                   NIB=NIB1                              X
                   AREA=FULLWORD,AREALEN=4
```

INQ1 determines whether the application program (whose symbolic name has been placed in NIB1's NAME field) is active and accepting logon requests. The answer is returned in FULLWORD.

```
INQ2     INQUIRE   RPL=RPL2,      RPL MODIFIERS FOLLOW   X
                   OPTCD=LOGONMSG,                       X
                   ACB=ACB1, NIB=NIB1,                   X
                   AREA=LGNMSG,AREALEN=(4)
```

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

Return of Status Information

When the INQUIRE operation is completed, these sources of status information can be checked:

The RECLEN field of the RPL: If INQUIRE completes its function normally, RECLEN contains the number of bytes of data that have been placed in the work area designated by the AREA field. If the FDBK field indicates that this data area was too small, RECLEN indicates the required length.

Register 15: One of the following hexadecimal values is indicated.

0 If the ASY option code is in effect, no error has been detected in the way the INQUIRE request was issued, and INQUIRE processing has been scheduled. If the SYN option code is in effect, INQUIRE processing has been completed successfully.

4 The request cannot be scheduled because the RPL is currently in use by another request. The RPL's FDBK field has not been set. If an active LERAD exit list routine is available, it has been invoked.

8 A logical error occurred; the FDBK field can be examined to determine which one it was. If an active LERAD exit list routine is available, it has been invoked. (This return code is possible only when the SYN option code is in effect.)

INTRPRET -- Interpret a Logon Message

The INTRPRET user supplies VTAM with a terminal's logon message and receives in return the identification of the application program implied in the logon message. ('Implied' here means that the terminal is not necessarily entering the application program's symbolic name in a form that is recognized throughout the teleprocessing network.)

When the installation is defining its teleprocessing network to VTAM, it identifies each terminal and associates with each a *logon characteristics table.* Each entry in the table contains several parts, but there are three that are of importance to the INTRPRET user:

A logon sequence.

An application program identification (or the address of a routine that will generate an application identification when executed).

An additional data sequence.

The *logon sequence* is a sequence of characters that the terminal might enter, presumably as the first part of a logon request (where the terminal is indicating the identity of the application program to which it wants to be connected). The logon sequence could be anything the terminal is capable of entering – graphic characters, a carriage return, or a function key. It amounts to an alias for the actual name of an application program.

The *application program identification* is the symbolic name of an application program. This is the name the installation assigns to an APPL entry when it is defining the application programs in its teleprocessing network.

The *additional data sequence* is simply any character string that the installation wants to associate with the logon sequence.

When a terminal sends an application program a logon message and that application program uses INTRPRET, VTAM looks for a matching logon sequence in that terminal's logon characteristics table. If a match is found, VTAM passes back the application program identification associated with the logon sequence. If the logon sequence also has an additional data sequence associated with it, the additional data is also passed back to the application program issuing INTRPRET.

The installation may choose not to use a terminal's logon characteristics table in the manner described above; routine addresses could be used in place of the application program's identification, for example. In any event, INTRPRET's function is the same: translate a given logon sequence into an application identification, and add an additional data sequence, if any was indicated by the installation.

The INTRPRET user supplies VTAM needed information via an RPL. The RPL fields described below can be set with the INTRPRET macro instruction itself. (This list does not include RPL fields whose effects are common to all VTAM macros; see the RPL macro instruction description, particularly Figure 5, for a list of all RPL fields applicable to INTRPRET) The RPL modifiers shown below have the following effects:

NIB=address
The NIB field must contain the address of a NIB; the NAME field of this NIB must contain the symbolic name of the terminal issuing the logon request. The symbolic name is provided in the parameter list supplied to the LOGON exit list routine.

AREA=address
VTAM obtains the logon message from this data area. The application program can obtain the logon message with the INQUIRE macro instruction or from the parameter list passed to the LOGON exit list routine; the application program then supplies this message to VTAM when it issues the INTRPRET macro instruction.

RECLEN=length
VTAM uses this value to determine how many bytes of data (pointed to in AREA) are to be used. This value is also provided in the LOGON exit list routine's parameter list, or it can be determined by using the INQUIRE macro instruction.

AAREA=address
INTRPRET places the information it produces into the work area indicated by the AAREA field. The information is arranged in this format:

The application program identification, as it exists in the logon characteristics table. (This is eight bytes long.)

The length of the additional data sequence that follows; if none follows, the indicated length is 0. (This length indicator is one byte long.)

The additional data sequence, if the installation included one in the logon characteristics table entry.

AAREALN=length
This field is set by the INTRPRET user to indicate the maximum length (in bytes) allocated for the work area indicated by the AAREA field.
After VTAM has placed the data into the work area indicated by AAREA, the RPL's ARECLEN field is set to indicate how many bytes were placed there. If the data to be placed in the AAREA work area is too long to fit (that is, exceeds the value contained in the AAREALN field), VTAM posts an error return code and indicates the required length in the ARECLEN field.

If the INTRPRET user does not know the required length for AAREA and wants to find this out dynamically (rather than allocate an area large enough to handle all cases), INTRPRET can be issued once with AAREALN set to 0, and then issued a second time with AAREALN set to whatever value the first INTRPRET returned in the ARECLEN field of its RPL.

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

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: 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 keyword=new value

Function: Indicates an RPL field to be modified and the new value that is to be contained or represented within it.

Format: For *rpl keyword* 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.

Note: All the RPL fields that have a unique effect on INTRPRET are discussed above. Check Figure 5 in the RPL macro instruction description for a list of all the RPL fields that are applicable for INTRPRET.

Example

Assume that an application program has obtained a terminal's logon message and wants to determine which application program the terminal is really indicating in its message. In this example, the application program has already obtained the length of the logon message (in register 3).

```
INT1        INTRPRET  RPL=RPL1,   RPL1 MODIFIERS FOLLOW:     X
                      NIB=NIB6,AREA=LGNMSG,RECLEN=(3)         X
                      AAREA=INT1DATA'AAREALN=40
```

When INT1 is issued, VTAM finds the logon characteristics table that the installation has defined for the terminal indicated by NIB6. VTAM next looks for the logon sequence in that table that matches the message provided in LGNMSG. When the entry is found, VTAM places the application identification associated with that logon sequence into INT1DATA. If an additional data sequence has also been included with the logon sequence, that data (and its length) also is placed into INT1DATA following the application identification. VTAM sets the RPL's ARECLEN field to indicate the total number of bytes actually brought into INT1DATA. Should the total amount exceed 40 bytes, INT1 will complete in error with the RPL's ARECLEN field set to the correct length.

Return of Status Information

When the INTRPRET operation is completed, these sources of status information may be checked:

The ARECLEN field of the RPL: If the FDBK field indicates that INTRPRET failed because the data to be placed in the AAREA work area would not fit, ARECLEN contains the number of bytes required to hold the data. If FDBK (or register 15) indicates that INTRPRET was completed successfully, ARECLEN indicates how many bytes of data have actually been placed in the AAREA work area.

The FDBK field of the RPL: If register 15 indicates that a logical error occured, this field indicates the type of logical error.

Register 15: One of the following hexadecimal values is indicated:

0 If the ASY option code is in effect, VTAM found no errors or contradictions in the INTRPRET request and has accepted the INTRPRET request. If SYN is in effect, the INTRPRET operation has been successfully completed.

4 The INTRPRET request cannot be accepted because the RPL is currently in use by another request. The RPL's FDBK field has not been set. The LERAD exit list routine, if an active one exists, has been invoked.

8 A logical error (see Appendix A) has occurred; the FDBK field of the RPL can be examined to determine which one it was. If an active LERAD exit list routine exists, it has been invoked. (This code can only be returned when the SYN option code is in effect.)

LDO -- Create a Logical Device Order

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 3270 display unit's buffer to the buffer of any printer or display unit attached via the same control unit.

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.)

Read a block of data from a 3735 terminal or from a System/3 or System/370 CPU, and then send a positive or negative acknowledgment accompanied by leading graphic characters, to the device.

Write data beginning with a block of heading characters to a 3735 terminal or to a System/3 or System/370 CPU.

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 that points to the LDO.

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 these parts:

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

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.

A *length* indicator. This indicates the length of the data or data area. (*Note:* The RPL also has corresponding data address and length fields, but these are ignored when the DO macro instruction is executed.)

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

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

symbol

Function: Provides a name for the macro instruction. This name can be used as the value of the AREA operand when the AREA field of the DO macro instruction's RPL is set to point to this LDO.

CMD=command

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

COPYLBM	WRITE	WRTNRLG
COPYLBT	WRITELBM	WRTPRLG
READ	WRITELBT	
READBUF	WRTHDR	

Function: Indicates the specific I/O operation to be performed when the LDO is used when the DO macro instruction is executed.

COPYLBM
This LDO causes VTAM to send the entire contents of a 3270 display unit's buffer to a printer or another display unit in the same remotely attached information display system. VTAM sends the data as a message by adding an ETX line control character at the end.

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 right-most two bytes of the *sending* device's CID. (See the copy command in the 3270 component description manual for an explanation of the copy control character.)

The ARG field of the DO macro instruction's RPL must contain the ***receiving*** 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 3735 programmable buffered terminal 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 to a WRTPRLG or WRTNRLG LDO. This allows the application program to either (1) send a negative acknowledgment to the device and then re-read 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 also send leading graphic characters along with the response.

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, error information is placed in the RPL's FDBK field, the required length is placed in the RPL's RECLEN field, and the LDO's address is placed in the RPL's AAREA field. The DO macro instruction in this situation, terminates with an error code posted in register 15.

READBUF
The READBUF (read buffer) LDO causes the entire contents of a 3270 display unit'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 the ENTER key).

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. If the data is too long to fit, error information is placed in the FDBK area of the RPL (see Appendix A), the required length is placed in the RECLEN field of the RPL, and the LDO's address is placed in the AAREA field. The DO macro instruction, in this situation, terminates with an error code posted in register 15.

WRITE

The WRITE LDO writes a block of data to a 3735 terminal or 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 to a 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

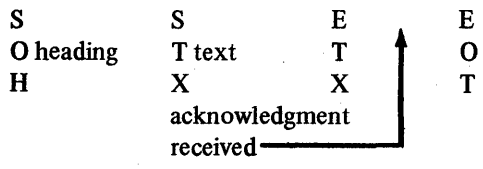
The WRITELBM LDO will write a block of data to a 3735 terminal or to a System/3 or System/370 CPU. For these devices, WRITELBM works exactly like a WRITE macro instruction with an LBM option code; an STX character is added to the beginning of the data, and an ETX character is added to the end. However, if a WRITELBM LDO is command-chained to 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 LDO writes a block of data to a 3735 terminal or 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 and followed with an STX and an ETX character respectively, and when an acknowledgment is received from the device, an EOT character is sent. However, if a WRITELBT LDO is command-chained to a WRTHDR LDO, this sequence is written:



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 3735 terminal or 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 command-chained to a WRTHDR LDO (by specifying FLAGS=C on the WRTHDR LDO), the ETB character is not inserted after the heading.

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 3735 terminal or 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 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 3735 terminal or 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.

ADDR=data address or data area address
Function: 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 the rightmost two bytes of the sending 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.

LEN=data length or data area length
Function: 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. If there is too much data, an error condition is raised and no data is placed there. For all output LDOs, LEN indicates how many bytes of data are to be written.

FLAGS=C

Function: 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 should only be used with the *WRTHDR* LDO (to command-chain it to a WRITE, WRITELBM, or WRITELBT LDO), or with the *WRTPRLG* or *WRTNRLG* LDOs (to command-chain them to a READ LDO). The absence of this operand indicates to DO that no further LDOs are to be used.

Examples

The following example illustrates the use of the COPYLBM LDO.

```
PRIME     SHOWCB  NIB=NIB1,AREA=TEMP,LENGTH=4,FIELDS=CID
          MVC     CPYSCRN1+1(2),TEMP+2
CPYSCRN   DO      RPL=RPL1,ARG=(6),AREA=LDO1
                          .
                          .
                          .
LDO1      LDO     CMD=COPYLBM,ADDR=CPYSCRN1,LEN=3
TEMP      DS      F         (TEMP=WORK AREA FOR CID)
CPYSCRN1  DC      X'630000' (63=A COPY CONTROL CHARACTER,
                            0000=FINAL AREA FOR RIGHT
                                 HALF OF CID)
```

The purpose of the two instructions at PRIME is to obtain the CID of a device (from NIB1 into TEMP) and place the right-most two bytes of the CID into a data area pointed to by LDO1. When CPYSCRN is executed, the device whose CID has been placed in register 6 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=LDOH
                          .
                          .
                          .
LDOH      LDO     CMD=WRTHDR,ADDR=AHDRBLOK,LEN=5,FLAGS=C
LDOT      LDO     CMD=WRITE,ADDR=ATXTBLOK,LEN=16
```

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

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

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.

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. MODCB can, however, be used to add valid addresses to dummy exit addresses and to change the A and N (active and inactive) attributes for those addresses.

An RPL cannot be modified while a connection or I/O 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 a modified for use with another access method.

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

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={ L | (E, parameter list address) }]**

symbol

Function: Provides a name for the macro instruction.

AM=VTAM

Function: Identifies this macro instruction as a VTAM macro instruction. This operand is required for the DOS/VS assembler, but is ignored by the OS/VS1 and OS/VS2 assemblers.

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

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

Format: Use only one of the four operands and supply the address of the control block. Register notation cannot be used in the list form.

Note: One of these operands must be selected for the standard or list form, but for the execute form none need be specified.

field name=new value

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

Format: 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 used. If RPL=RPL1 is coded, for example, the keyword of any operand in the RPL macro instruction can be coded.

For *new value* indicate a register or code any value that could be used were the operand specified in an ACB, EXLST, RPL, or NIB macro instruction. Expressions involving registers cannot be used in the list form.

When "field name" is an EXLST keyword (LERAD, TPEND, etc.), the A or N attributes and the E, LE, or GE attributes can be specified without specifying the address also.

Examples:

```
LERAD=(,N)
TPEND=(,A,LE)
```

Note the use of the commas and parentheses.

Note: The standard and list forms of MODCB must indicate at least one field to be modified. The execute form does not require this operand.

[, MF= { L | (E, parameter list address) }]

Function: Indicates that either a list form or an execute form of MODCB is to be used.

MF=L
The list form (L-form) of this macro instruction creates a parameter list for later use by the execute form. Because the L-form macro instruction generates only this parameter list, and no executable code, operand forms like register notation are prohibited. Only relocatable expressions valid for adcons can be used. The user is responsible for branching around the generated parameter list, which is variable in length.

MF=(E,parameter list address)
The execute form (E-form) of this macro instruction can modify the parameter list generated by its list form, and causes control to be passed to VTAM routines when the E-form is executed. The expansion of the execute form provides the executable instructions required to perform parameter list modification and passing of control.

The *parameter list address* should specify the location of the list form of the macro instruction. Expressions involving registers can be used.

Note: Although the execute form of MODCB can modify the list form's parameter list, it cannot add to it. Therefore if an operand value will not be known until program execution and is to be supplied with the execute form. the list form must also specify that operand and supply a dummy value for it.

For example: Assume that during program execution the address of a LERAD exit routine will become available in register 3. To replace the LERAD entry in the exit list (in this case EXLST1), code a list form like this:

```
LOOD    MODCB    EXLST=0,LERAD=0,MF=L
```

and code an execute form like this:

```
EMOD    MODCB    EXLST=EXLST1,LERAD=((3),A),MF=(E,LMOD)
```

Example

```
MOD1    MODCB    RPL=(5),OPTCD=(ASY,SPEC,LBM)
```

MOD1 activates the ASY, SPEC, and LBM option codes in an RPL. The address of this RPL must be in register 5 when MOD1 is executed.

Return of Status Information

After MODCB processing is completed, VTAM indicates one of the following hexadecimal values in register 15 before passing control back to the application program:

- 0 All of the control block fields were successfully modified.
- 4 The location indicated by the ACB, EXLST, RPL, or NIB operand does not contain a valid control block.
- 8 The execute form was used incorrectly. For example, a control block field was being modified, but the list form did not supply a dummy value for it.
- C A field to be modified is not a control block field for which modifications are allowed. Only those fields whose contents can be set by ACB, EXLST, RPL, and NIB macro instructions can be modified.

NIB--Create a Node Initialization Block

The NIB generated by the NIB macro instruction is used by the program to identify which terminal 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 transaction 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 represented by the NIB with other information contained there -- information that is placed in the NIB by the USERFLD and PROC operands of the NIB macro instruction. This association continues as long as the terminal remains connected. If the USERFLD or PROC information is to be altered during that time, the MODCB macro instruction must be used to make the appropriate changes in the USERFLD and PROC fields of the NIB, and the CHANGE macro instruction must be used to make these modifications effective.

NIBs can be grouped together into lists. When certain 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 NIB can be built during program assembly with the NIB macro instruction, or it can be built during program execution with the GENCB macro instruction.

A field called the CID field is generated as 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 NIB is connected to the program, VTAM generates a shortened version 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. Subsequent I/O requests directed toward that specific terminal must have this CID in the I/O request's RPL. If the one placed in the RPL during connection is lost, the SHOWCB macro instruction can be used to obtain the one placed in the NIB.

Name	*Operation*	*Operands*
[symbol]	NIB	[NAME=name in resource definition table] [, USERFLD=fullword of terminal data] [, LISTEND={YES|NO}] [, MODE=BASIC] [, PROC= ([, {BLOCK|MSG|TRANS|CONT}] [, {LGOUT|NLGOUT}] [, {CONFTXT|NCONFTXT}] [, {TMFLL|NTMFLL}] [, {EIB|NEIB}] [, {TIMEOUT|NTIMEOUT}] [, {ERPIN|NERPIN}] [, {ERPOUT|NERPOUT}] [, {MONITOR|NMONITOR}] [, {ASYIPX|NASYIPX}] [, {ELC|NELC}] [, {TRUNC|KEEP}] [, {BINARY|NBINARY}])]

symbol

Function: Provides a name for the macro instruction and thus for the NIB generated by it. This name can be used for the ARG operand of an RPL macro instruction.

NAME=name in resource definition table

Function: Associates the NIB with a terminal represented in the resource definition table. When used by the INQUIRE macro instruction with OPTCD=APPSTAT, the NAME field associates the NIB with an application program represented in the resource definition table. (The resource definition table is built by the installation during VTAM definition.)

Format: Use the name of the TERMINAL, COMP, LOCAL, or APPL entry that represents the terminal or application program in the resource definition table. Code this name as it appears in the resource definition table. For example:

```
NAME=TERM13
```

Note: Although this operand is optional, the NAME field should be set by the time the OPNDST 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.

USERFLD=fullword of terminal data

Function: Indicates any fullword of data that the application wants to associate with the terminal represented by this NIB.

When the terminal is connected (and also whenever a READ macro instruction with an ANY option code is issued), VTAM places the contents of the USERFLD field in the USER field of the RPL being used for the connection or read request.

The fullword of data can be anything the application 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 READ macro with OPTCD=ANY, since the setting of the USER field in the READ macro's RPL provides the only efficient way the program can establish the identity of the terminals from which the data was just obtained. Or the fullword could be the address of a routine that the application program invokes whenever it connects the terminal.

Format: Code the fullword of data in either character 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'TERM1', USERFLD= X'00043E0', USERFLD=A(RTN1), and USERFLD=V(RTN2).

Note: 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.

LISTEND={ YES\|NO }

Function: Allows the application program to group NIBs into lists. LISTEND=YES indicates that this NIB is the last in a list. LISTEND=NO indicates that this NIB and the NIB immediately following it in storage are part of a NIB list.

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

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 simply a list of one.

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

MODE=BASIC

Function: Before an application program can issue an OPNDST or CHANGE macro instruction for a NIB, that NIB's MODE field must be set to BASIC. (BASIC allows VTAM to distinguish between NIBs used by the application program and NIBs used by VTAM itself.)

This operand is optional, but if MODE=BASIC is not specified in the NIB macro instruction you will have to specify it in a MODCB macro instruction before OPNDST or CHANGE are executed.

PROC={ processing option (processing option,...)}

Function: Indicates options VTAM is to follow for all I/O requests involving the terminal associated with this NIB.

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

```
NIB         NAME=TERM13,MODE=BASIC,                          X
            PROC=(BLOCK,NERPIN,NERPOUT)
NIB         NAME=TERM14,MODE=BASIC,                          X
            PROC=BLOCK
```

Note: Not all processing options are valid for all types of devices. See Figure 4 at the end of this macro instruction description to see which processing options are valid for any given device supported by VTAM.

To change any of the processing options after OPNDST has been issued, follow this two-step procedure:

1. Issue a MODCB macro instruction. In this macro indicate the appropriate NIB and use the PROC operand to specify the new processing options.
2. Issue a CHANGE macro instruction. In this macro indicate the RPL that in turn points to the NIB whose processing options were modified by the MODCB macro.

{**BLOCK**|**MSG**|**TRANS**|**CONT**}
These control how many blocks of data are to be obtained from a terminal for a solicit operation and how acknowledgments (responses) are to be handled as each block arrives.

Solicit operations are all operations conducted by VTAM to obtain data from a terminal and transfer it 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 3. 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
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. A line control response is sent to acknowledge receipt of the data obtained from the *previous* solicit operation, but no such 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 binary synchronous device, an installation 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 thruput that is very low compared to MSG, TRANS, and CONT.)

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 recognized. In effect, this means that data is solicited

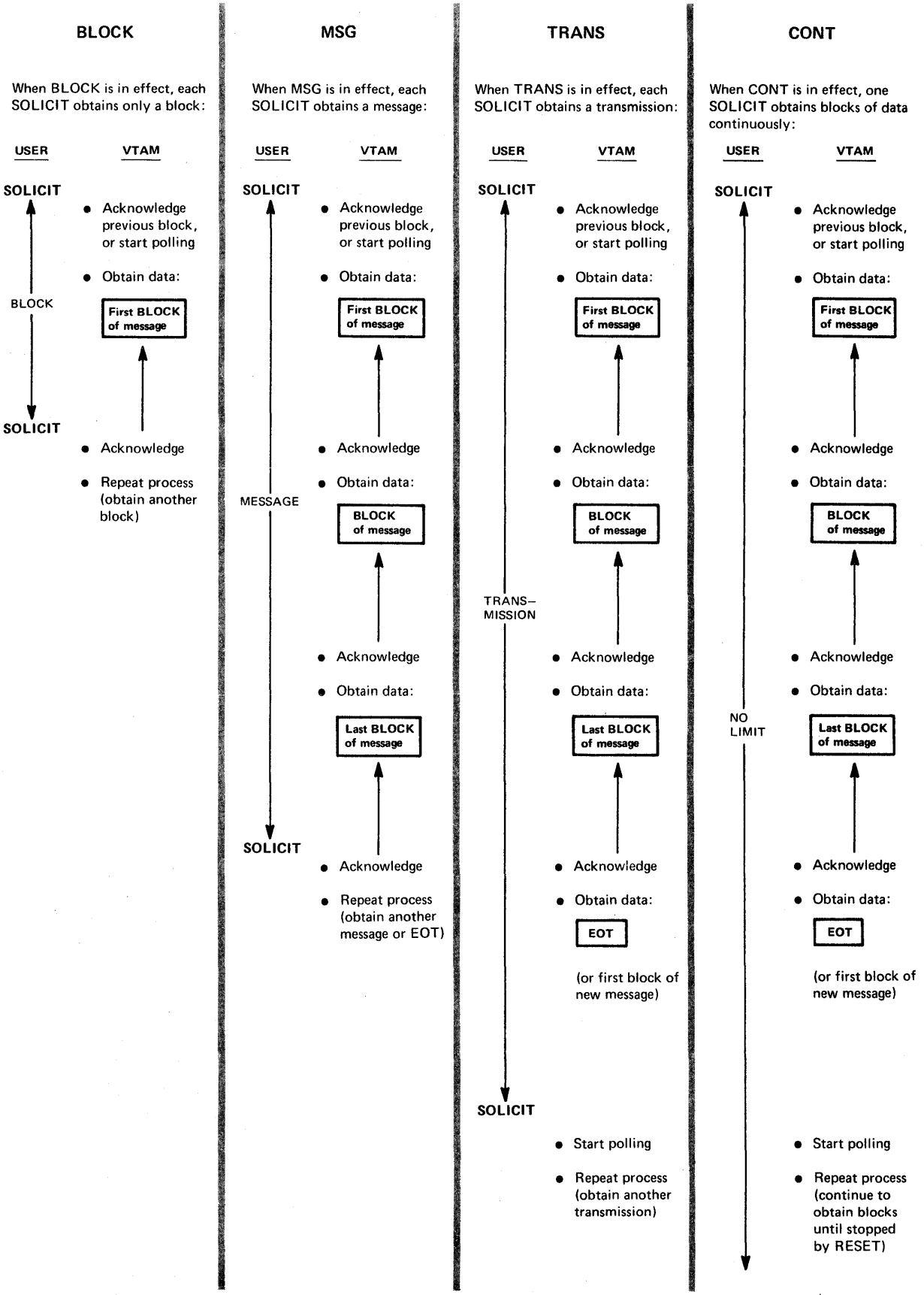


Figure 3. The Effect of BLOCK, MSG, TRANS, and CONT on Solicitation

from the terminal until an entire message has been received. 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.

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. Line control responses are sent as each block is received, including the last block. Polling will not resume 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**}
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 includes leading graphic characters. When LGOUT is specified, a special 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 will cause 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 will be passed to the application program as the input data.

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

{**CONFTXT**|**NCONFTXT**}
Indicates whether or not the data sent to or received from this terminal is to be considered as 'confidential.' If CONFTXT is specified, VTAM clears any of its buffers used to hold the data before returning them to operating system buffer pools. For NCONFTXT, no such clearing will be done.

{**TMFLL**|**NTMFLL**}
Indicates whether or not the system is to insert idle device control characters or equivalent time-fill device-control characters into the data sent to this terminal. TMFLL allows the 3704 or 3705 communications controller to insert these characters. * NTMFLL suppresses the system's insertion of these characters -- implying that the application program will be supplying its own time-fill characters.

{**EIB**|**NEIB**}
Indicates whether or not the system is to insert an EIB (error information byte) after every ITB character received from this terminal. EIB indicates that an EIB is to be inserted with each intermediate transfer block; NEIB suppresses the insertion of EIBs.

*For information regarding the characters normally inserted see the *IBM 3705 Communications Controller Network Control Program, Generation and Utilities Guide and Reference Manual,* GC30-3000.

{**TIMEOUT**|**NTIMEOUT**}
Indicates whether or not the 3704 or 3705 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 3704 or 3705 imposes a text timeout limitation if the installation so indicated 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.

{**ERPIN**|**NERPIN**}
Indicates whether or not system error recovery 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.

{**ERPOUT**|**NERPOUT**}
Indicates whether or not system error recovery 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.

{**MONITOR**|**NMONITOR**}
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 VTAM 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 installation indicated during VTAM definition that the 3704 or 3705 communications controller is to react to attention interruptions. If an attention interruption is received *during,* an I/O operation, the I/O request fails with the RPL FDBK 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.

{**ASYIPX**|**NASYIPX**}
Indicates the action VTAM is to take when data arrives in VTAM buffers from this terminal as a result of a solicit request. If ASYIPX is specified, the ASYIP exit list routine (see EXLST macro) is invoked provided that the CA option code is also in effect for the solicit request. If NASYIPX is specified (or if the CS option code is in effect), the ASYIP exit list routine is not invoked. The data is obtained with a READ macro, regardless of whether the ASYIP exit list routine is invoked or not.

{**ELC**|**NELC**}
Indicates whether line-control characters are to be generated for the data sent to this terminal. ELC signifies that the application is embedding its own line 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 line control characters. See Appendix B for a list of the line control characters that are normally inserted by VTAM. ELC can only be used if the NBINARY option code is in effect for the RPL.

{BINARY|NBINARY}

Indicates how data is to be handled when a WRITE macro instruction is used to write to a binary synchronous device.

BINARY
The data is sent in transparent text mode. This means that all of the data in the program storage area is transmitted to the terminal only as bit patterns. Any bit patterns can therefore be sent, such as EBCDIC, control characters, or object code.

NBINARY
The data is not sent in transparent text mode.

{TRUNC|KEEP}

Indicates the action VTAM is to take when data received from the terminal as a result of a READ macro instruction is too long to fit in its input area.

TRUNC
The excess data is truncated and lost, and the read operation is completed with a physical error. This invokes the SYNAD exit list routine, if an active one exists.

KEEP
VTAM saves the excess data; no error condition results, but in the FDBK field of the READ macro's RPL, VTAM indicates what happened. The next READ macro instruction directed at the terminal will retrieve the excess data.

Example

```
NIB      NIB      NAME=KBD3270,USERFLD=A(KBDRTN),      X
                  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. Before OPNDST processing is completed, VTAM obtains the contents of NIB1's USERFLD field (which in this example is the address of a routine) and places it in the USER field of the OPNDST's RPL. Note that PROC has not been specified in the NIB1 macro instruction; all the underscored values for PROC (as shown in the assembler format table above) will be associated with the terminal until MODCB and GENCB macro instructions are issued for NIB1.

Devices Applicable for Each NIB Processing Option

The following figure shows the processing options applicable to each device supported by VTAM.

An X indicates that the PROC operand value can be used with the device. As the assembler format table for NIB shows, all of the PROC operand values occur in pairs, with the exception of the BLOCK-MSG-TRANS-CONT option. This table shows only one of each of these pairs, since the other half is valid for *all* devices supported by VTAM.

Start–Stop Devices:	BLOCK	MSG	TRANS	CONT	LGOUT	CONFTXT	NTMFLL	EIB	NTIMEOUT	NERPIN	NERPOUT	MONITOR	ASYIPX	ELC	TRUNC
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	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
AT&T Teletypewriter Terminal, Models 33 and 35			X	X		X	X		X				X	X	X
Western Union Plan 115A Station			X	X		X							X	X	X
Binary Synchronous Devices:															
IBM 2770 Data Communication System	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
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			X			X							X		X
IBM 3270 Information Display System, remotely attached to controller			X			X							X		X
IBM 3735 Programmable Buffered Terminal	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
IBM 3780 Data Transmission Terminal	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
IBM SYSTEM/370	X	X	X	X	X	X		X		X	X		X	X	X

Figure 4. Devices Applicable to each NIB Processing Option

OPEN -- Open one or more ACBs

The purpose of the OPEN macro is to open (or "activate") the ACB so that the ACB and all subsequent requests directed to it can be identified by VTAM as representing a specific application program. Accordingly, 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 a symbolic name. This symbolic name is generated during VTAM definition by the installation when it defines the application program; the entry is generated with the APPL definition macro instruction, and is called an APPL entry. It is during OPEN processing that the association between the ACB and the APPL 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.

Should the APPL entry contain a password, the ACB being opened must specify that same password, or OPEN will not be completed successfully.

When MACRF=LOGON is specified for the ACB being opened, OPEN also serves to notify VTAM to queue any logon requests directed to the APPL entry associated with the ACB.

OPEN generates logon requests on behalf of terminals if the installation has so indicated during VTAM definition. (These logon requests are called *automatic logon requests.)*

If the ACB's EXLST operand was used, VTAM can associate the ACB with the exit list indicated by the EXLST operand. The exit list (generated by the EXLST macro instruction) contains the addresses of routines to be invoked when specific types of events occur. Once OPEN processing has been completed, VTAM knows which exit list should be used when one of these events occur.

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/VS1 and OS/VS2 only.		

symbol

Function: Provides a name for the macro instruction.

acb address

Function: 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/VS1 or OS/VS2. (The same macro expansion program services 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/VS1 and OS/VS2 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 fifteen addresses.

Example

```
OPEN123  OPEN  ACB1,ACB2,(7)
```

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 contains a code indicating whether or not the OPEN processing was completed successfully. Successful completion means that *all* ACBs specified in the OPEN macro instruction have been 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; unsuccessful completion is indicated by a return code other than 0.

If unsuccessful completion is indicated, the application must 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
```

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 macro instruction. For example:

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

The hexadecimal contents of the ERROR field indicates the error encountered for that ACB during OPEN processing:

- 00 OPEN has successfully opened this ACB.
- 04 An OPEN macro instruction has already been successfully issued for this ACB.
- 24 The password specified in the ACB does not match the password specified in the corresponding APPL entry, or the ACB does not specify a password when one should have been specified.
- 50 VTAM has not been included as part of the operating system.
- 52 VTAM has been included as part of the operating system, but the network operator has issued a HALT command, and VTAM is shutting down.
- 54 The OPEN macro instruction was issued with a syntax error -- such as an omitted comma.
- 56 A match for the APPLID was found in the resource definition table, but it was for an entry other than an APPL entry.

58 Another ACB, already opened by VTAM, indicates the same APPLID that this ACB does.

5A No entry could be found in the resource definition table that matches the name supplied in the ACB's APPLID field.

5C VTAM has been included as part of the operating system, but VTAM is not yet active.

5E The APPLID field points to a storage area that is not in your program area.

62 The length indicator in the field pointed to by the APPLID field exceeds the allowable limit (8).

64 The PASSWD field points to a storage area that is not in your program area.

66 The length indicator in the field pointed to by the PASSWD field exceeds the allowable limit (8).

OPNDST -- Establish Connection with Terminals

The OPNDST (open destination) macro instruction requests for VTAM to establish connection between the application program and one or more terminals.

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.

- An application program can simply request that a terminal be connected to it. Such a request is satisfied as soon as the terminal is available -- that is, has been activated but has not issued a logon request. If the terminal is connected to another application program, however, the terminal cannot be reconnected until that application program releases it. (The other application program is notified of your request via the invocation of its RELREQ exit list routine, and it releases the terminal by issuing a CLSDST macro instruction.) OPNDST can succeed in reconnecting the terminal only when -- or if -- the other application program chooses to release the terminal.

  If a terminal has been defined as a dial-in terminal by the installation (CALL=IN specified for the LINE or GROUP definition macro), a connection request is completed when the terminal operator dials in. If a terminal has been defined as a dial-out terminal by the installation (CALL=OUT), the connection request is completed immediately, but the terminal is dialed only when an I/O request is issued for the terminal.

  This type of request is implemented by setting the ACQUIRE option code in the RPL used by OPNDST. For application programs running under OS/VS1 or OS/VS2, the use of ACQUIRE must be authorized by the installation.

- 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.

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

  This type of request is implemented as indicated above, except that the ACCEPT option is set in the RPL.

Note: When a terminal requests logon, VTAM first checks for an active LOGON exit list routine -- and invokes the routine if any active one is found. If no active routine exists, VTAM checks for outstanding OPNDST requests (that is, OPNDSTs with ACCEPT that have not yet been completed because no logon request has been made by the terminal). Thus a logon request will not cause a pending OPNDST with ACCEPT to complete if an active logon exit list routine is available.

There are two versions of OPNDST with OPTCD=ACQUIRE. These versions are specified with two more RPL options, CONALL and CONANY, which govern whether an entire group of terminals is to be connected, or whether any single terminal in this group is to be connected. OPNDST with ACCEPT likewise has two

versions, specified with two RPL option codes -- SPEC and ANY. These govern whether a specific terminal is to be connected or whether any eligible terminal is to be connected.

There are therefore a total of four types of OPNDST. The following four sections indicate how you must prepare for each and what happens when the connection occurs.

OPNDST with ACQUIRE and CONALL options

You must do this:

Set the RPL's NIB field to point to a *list* of NIBs (described in the LISTEND operand of the NIB macro instruction). The ACQUIRE and CONALL option codes must be set in the RPL.

And when OPNDST is issued, the result will be this:

VTAM connects the program to *all* of the terminals represented in the NIB list that have not issued logon requests, and:

Generates a CID for each connected terminal. Each CID (which is a shortened form of the terminal's symbolic name) is placed in its respective NIB in a field called the CID field, where it can later be obtained with the SHOWCB macro instruction when it is needed. Unlike other forms of OPNDST, the CID is *not* placed in the RPL's ARG field.

Places the CID of the last terminal in the ARG field of the RPL. I/O requests to specific terminals must have the CID of that terminal in the ARG field of the request's RPL. Thus the RPL used for OPNDST can conveniently be used for subsequent I/O requests with a terminal. If the CID placed in the RPL is lost, its copy that was placed in the NIB can be retrieved and used. *Note:* The RPL's NIB and ARG fields actually occupy the same physical field. There is more information on this in the description of the RPL macro's NIB operand.

Places the address of the first NIB of the NIB list in the AREA field of the RPL (This is the same address you specified for the RPL's NIB field; it is returned because the contents of the NIB field may have been destroyed during connection.)

Extracts the contents of the USERFLD field from a NIB (the NIB associated with the last terminal connected) and places it in the USER field of the RPL.

OPNDST with ACQUIRE and CONANY options

You must do this:

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

And when OPNDST is issued, the result will be this:

VTAM connects the program to the *first* (and only to the first) available terminal represented in the NIB list. A terminal is available if it is not connected to any application program, and has not issued a logon request. 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 associated with it.

Sets a flag in the NIB that represents the connected terminal. The application program can locate this NIB by issuing a TESTCB macro (with a CON=YES operand) for each NIB.

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 NIB field; it is returned because the contents of the NIB field are destroyed when the CID is placed in the ARG field.)

Places the contents of that NIB's USERFLD field into the USER field of the RPL.

OPNDST with ACCEPT and ANY Options

You must do this:

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 basic.

And when OPNDST is issued, the result will be this:

VTAM connects the program to any terminal that has directed a logon request 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.

Places the contents of the USERFLD field of the NIB in the USER field of the RPL.

OPNDST with ACCEPT and SPEC options

You must do this:

Set the RPL's OPTCD field to ACCEPT and SPEC, and set the NIB field to point to a NIB.

And 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 request 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.

Places the contents of the NIB's USERFLD field into the USER field of the RPL.

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 5 in the RPL macro instruction description for a list of these codes and fields.

The one exception is the Q-NQ option code. An OPNDST request with ACQUIRE establishes connection with a terminal that is connected to another application

program only when that application program chooses to release it. An OPNDST issued with the Q option code completes only if the terminal becomes available. If NQ is used instead, VTAM cancels the request and returns control immediately to the application program if the terminal is not available.

OPNDST with ACQUIRE in combination with the Q option code should therefore be used judiciously, and then only if the connection request is specified as asynchronous (ASY option code). Synchronous OPNDST requests with the ACQUIRE and Q processing options are not permitted. (The reason: If application program A issues this kind of connection request and the terminal is connected to application B, application A cannot receive control back until application B releases the terminal. If application B, before releasing the terminal, makes the same kind of request for a terminal connected to application program A, neither application program can regain control.)

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

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

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

rpl keyword=new value

Function: Indicates an RPL field to be modified and the new value that is to be contained within it.

Format: For *rpl keyword* code the keyword of the RPL macro instruction operand that corresponds to 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.

Note: All of the RPL fields that have a unique effect on OPNDST (and thus might be modified here) are discussed above. Check Figure 5 in the RPL macro instruction description for a list of all the RPL fields that are applicable for the OPNDST macro instruction.

Example

```
ACQALL  OPNDST  RPL=RPL1,      THIS IS AN "ACQUIRE & CONALL"  X
                ARG=NIBLIST1,  OPNDST: CONNECT ALL TERMS.     X
                ACB=ACB1       OF NIBLIST1  TO ACB1.


ACQANY  OPNDST  RPL=RPL2,      THIS IS AN "ACQUIRE & CONANY"  X
                NIB=NIBLIST2,  OPNDST: CONNECT A TERMINALX
                ACB=ACB1       OF NIBLIST2  TO ACB1.
```

```
ACPTANY OPNDST  RPL=RPL3,     THIS IS AN "ACCEPT & ANY"      X
                NIB=NIB6,       OPNDST: CONNECT ANY          X
                ACB=ACB1,       ELIGIBLE TERMINAL...         X
                OPTCD=ANY       ...TO ACB1.

ACPTSPC OPNDST  RPL=RPL4,     THIS IS AN "ACCEPT & SPEC"     X
                NIB=NIB7,       OPNDST: IF ELIGIBLE, CONNECTX
                ACB=ACB1,       NIB7 TERMINAL...TO ACB1.     X
                OPTCD=SPEC
```

Return of Status Information

After the OPNDST operation is completed, the CID field of the NIB, and the ARG, USER, and AREA fields of the RPL are posted as indicated above. Register 15 indicates one of the following hexadecimal values:

- 0 If the ASY option code is in effect, VTAM accepted the connection request. If the SYN option code is in effect, the connection (or connections) have been successfully established.
- 4 The request cannot be accepted because the RPL is currently in use by another request. The RPL's FDBK field has not been set. If an active LERAD exit list routine is available, it has been invoked.
- 8 A logical error occurred; the FDBK field of the RPL can be examined to determine which one it was. If an active LERAD exit list routine is available, it has been invoked. (This return code is possible only when the SYN option code is in effect.)
- C A physical error occurred; the FDBK field can be examined to determine which one it was. If an active SYNAD exit list routine is available, it has been invoked. (This return code is possible only when the SYN option code is in effect.)
- 10 The NQ option is in effect for the connection request, and the terminal to be connected is not available.
- 1C VTAM canceled the connection request; the second byte of the FDBK field is set indicating the reason.

READ -- Read Data into Program Storage

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 solicited from the terminal into program storage.

If READ is being used to obtain data from a specific terminal -- which means that the SPEC option code is in effect in the RPL -- and no data has been solicited or is being solicited from that terminal, READ first causes this data to be solicited. This implied solicit operation works in the same manner as the solicit operation explained in the SOLICIT macro instruction description.

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 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 solicited terminal the data is to be obtained, the location of the data 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 following list shows the effect of the more significant of these fields. See the RPL macro instruction description for a list and explanation of all RPL fields applicable for READ. The RPL modifiers shown below have the following effects:

ARG=(register)
If data is to be read from a specific terminal, the ARG field of the RPL must contain the CID for that terminal (see the OPNDST macro for an explanation of the CID). ARG=(register) is indicated here because that is the only way the CID can be placed in the ARG field with this READ macro instruction. (The CID can be extracted from the NIB with SHOWCB and then loaded into a register.)

If data is to be read from a specific terminal, the ARG field of the RPL must contain the CID for that terminal (see the OPNDST macro for an explanation of the CID). ARG=(register) is indicated here because that is the only way the CID can be placed in the ARG field with this READ macro instruction. (The CID can be extracted from the NIB with SHOWCB and then loaded into a register.)

If data is to be read from *any* solicited terminal, however, the ARG field's content is irrelevant when READ is issued. After the data has been read, VTAM sets the ARG field with the CID of the terminal from which the data originated.

AREA=address
The AREA field must contain the address of the data area in the program where the data is to be placed. Once the data has been moved, the RPL's RECLEN field is posted with the number of bytes that were placed there.

AREALEN=length
The AREALEN field must contain the length of AREA, in bytes. This value is used by VTAM to determine whether the incoming data is too long to fit. It it is too long, the action indicated by the TRUNC-KEEP processing option is taken.

OPTCD={CS|CA}
When the CA option code is in effect, there is no restriction on subsequent retrieval of data from the terminal that is the object of this READ macro instruction.

When CS is in effect, however, any subsequent solicit or read operation will exclude that terminal from the group of terminals eligible for solicit or read operations. This exclusion applies only if the ANY option code is in effect for the subsequent operation.

OPTCD={SPEC|ANY}
When the SPEC option code is in effect, data is obtained from a specific terminal 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 merely moves the data from the buffers to program storage.

When the ANY option code is in effect, only data already solicited 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 program. VTAM obtains the CID of the terminal from which the data originated and places it in the ARG field of the RPL.

The following option determines how excess data is to be handled (note that this is a NIB processing option, not an RPL option code):

PROC={TRUNC|KEEP}
When the TRUNC processing option is in effect and the incoming data is too large to fit in the storage area indicated by the AREA field, the data is truncated, the excess is lost, and the read operation terminates with an I/O error indicated.

Should the KEEP processing option be in effect instead, and the data is too long to fit, the excess is held for the time being and moved into the storage area when the *next* read request is issued.

Name	*Operation*	*Operands*
[symbol]	**READ**	**RPL=rpl address** **[, rpl keyword=new value]...**

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL that governs the read operation.

rpl keyword=new value

Function: Indicates an RPL field to be modified and the new value that is to be contained or represented within it.

Format: For *rpl keyword* 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.

Note: All of the RPL fields that have a unique effect on READ (and thus might be modified here) are discussed above. Check Figure 5 in the RPL macro instruction description for a list of all the RPL fields that are applicable for READ.

Examples

```
READ1    READ    RPL1,AREA=INFO,AREALEN=132                X
                 OPTCD=(ANY,SYN)
```

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.

```
SHOW1    SHOWCB  NIB1,FIELDS=CID,WAREA=TERMID,LENGTH=4
LOAD     L       3,TERMID
READ2    READ    RPL=RPL1,ARG=(3),AREA=INFO,               X
                 AREALEN=132,OPTCD=(SPEC,SYN)
```

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 termianl was placed both in NIB1 and in the RPL used for the connection macro (OPNDST). This example assumes that NIB1 has been left intact since then, but that the RPL has been reused and the CID originally placed there has been lost. It therefore is necessary to extract the CID from NIB1 and place it into a work area with SHOW1, load it into register 3 (with LOAD), and specify that register with READ's ARG operand.

Return of Status Information

Once the operation is completed, these sources of status information may be checked.

The RECLEN field of the RPL: RECLEN contains the number of bytes of data that were placed in program storage.

The ARG field of the RPL: If the ANY option code is in effect, ARG contains the CID of the terminal from which the data originated.

The USER field of the RPL: When a NIB is established, the user has the option of specifying any arbitrary 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 FDBK field of the RPL: Unless a value of 4 was returned in register 15, the FDBK (feedback) field may indicate error or completion status information, including whether the data just read was the last block of a message or an EOT. (See Appendix A for a description of the feedback field and the conditions under which it is set.)

Register 15: One of the following hexadecimal values is indicated:

0 If the ASY option code is in effect, VTAM found no errors or contradictions in the read request itself, and has accepted the read

operation. If SYN is in effect, the read operation has been successfully completed.

4 The request cannot be accepted because the RPL is currently being used by another request, or the terminal is not connected to your application program. The RPL's FDBK field has not been set. The LERAD exit list routine, if an active one exists, has been invoked.

8 A logical error (see Appendix A) occurred; the FDBK field can be examined to determine which one it was. If an active LERAD exit list routine is available, it has been invoked. (This code can only be returned when the SYN option code is in effect.)

C A physical error occurred; the FDBK field can be checked to determine which one it was. If an active SYNAD exit list routine is available, it has been invoked. (This code is also possible only when the SYN option code is in effect.)

14 A special condition exists -- for example, an attention interruption has been detected. The second byte of the FDBK field can be examined to determine the specific condition.

18 The READ operation was canceled by a RESET request.

1C VTAM canceled the READ operation; the second byte of the FDBK field is set indicating the reason.

RESET -- Cancel an I/O Operation

The RESET macro instruction can be used to:

Cancel an I/O operation that is pending, but is not in the process of being completed (that is, no data transfer activity has yet begun).

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.

Merely reset any error lock that may have been set for the terminal.

Note: The I/O operation most likely to be pending at any given time is a solicit operation. Once a terminal has been polled or otherwise readied, the solicit operation is pending until that data is forthcoming - which could be indefinitely.

The user of the RESET macro instruction uses an RPL option code to select one of the three variations of RESET:

OPTCD=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 FDBK field is set to indicate that cancellation did not occur. If a read or write operation is pending, that operation is posted as complete, and both register 15 and the FDBK field of that request's RPL indicate that RESET caused the premature completion of the operation.

OPTCD=COND also causes RESET to perform the same resetting operation indicated below under OPTCD=LOCK. OPTCD=COND is appropriate for situations in which the application program wants to write to a terminal only if no data is currently being sent to it (and can tolerate a resulting delay).

OPTCD=UNCOND
RESET cancels any I/O operation, pending or otherwise, that is being performed with the terminal. 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 complete, and the FDBK field of its RPL indicates 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 for situations in which 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=LOCK
RESET resets an error lock that has been set for the terminal. Error locks are set by a 3705 communications controller when it determines that it should not or can not continue to communicate with a terminal. Common reasons for the setting of error locks include:

Unrecoverable hardware malfunction.

Negative polling limit exceeded.

Attention interruption issued by the terminal.

Note: This type of RESET must not be used to cancel operations initiated by a DO macro instruction involving more than one LDO. Use RESET with OPTCD= UNCOND instead.

The address of an RPL must be supplied when the RESET macro is coded, and the ARG field of this RPL must contain the CID for the terminal whose I/O operation is to be canceled (or whose error lock is to be reset). See the OPNDST macro instruction for an explanation of the CID.

The ECB and EXIT fields of the RPL, and some of the option codes of the OPTCD field, govern actions to be taken then the RESET macro instruction is executed. The effect of these fields on RESET is the same as their effect on any I/O request macro instruction. See the RPL macro instruction for a description of these fields.

Name	*Operation*	*Operands*
[symbol]	**RESET**	**RPL=rpl address** **[, rpl keyword=new value]...**

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL that governs the execution of the RESET macro instruction.

rpl keyword=new value

Function: Indicates an RPL field to be modified and the new value that is to be contained or represented within it.

Format: For *rpl keyword* 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.

Note: All the RPL fields that have a unique effect on RESET (and thus might be modified here) are discussed above. Check Figure 5 at the end of the RPL macro instruction description for a list of all the RPL fields that are applicable for RESET.

Example

```
RESET1    RESET    RPL=RPL1,ARG=(3),ECB=ECBWORD,          X
                   OPTCB=(ASY,UNCOND)
```

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, these sources of status information may be checked.

The USER field of the RPL: When a NIB is established, the user has the option of specifying any arbitrary value in the USERFLD field of that NIB. When the RESET

macro instruction is subsequently issued to cancel I/O operations with 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.

The FDBK field of the RPL: Unless a value of 4 is returned in register 15, the FDBK (feedback) field may indicate error or completion status information regarding the execution of the RESET macro instruction. (See Appendix A for a description of the feedback field and the conditions under which it is set.)

Register 15: One of the following hexadecimal values is indicated:

0 If the ASY option code is in effect, VTAM found no errors in the way RESET was issued that would guarantee the eventual failure of the reset operation, and so has accepted the request. If the SYN option code is in effect, the reset operation has been completed successfully (that is, an I/O operation has been canceled).

4 The request cannot be accepted because the RPL is currently in use by another request, or the terminal for which RESET was issued is not connected to your application program. The RPL's FDBK field has not been set. If an active LERAD exit routine exists, it has been invoked.

8 A logical error (see Appendix A) occurred; the FDBK field can be examined to determine which one it was. If an active LERAD exit list routine is available, it has been invoked. (This code can be returned only when the SYN option code is in effect.)

10 A conditional reset operation was requested (OPTCD=COND), but an I/O operation had already reached the data-transfer stage. The I/O operation was therefore not canceled. (This code can only be returned when the SYN option code is in effect.)

1C VTAM canceled the RESET operation; the FDBK field is set indicating the reason.

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 READ 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 complete, and indicates a variety of other options to be followed while the request is being processed.

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. 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.

Requests for RPL modification can be made not only as part of a connection or I/O macro, but also by the MODCB macro instruction. Either way involves naming an RPL field and specifying its new content. It is useful to keep in mind that each 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.

The RPL macro instruction builds an RPL during assembly. An RPL can also be generated during program execution with the GENCB macro instruction. See GENCB for a description of this facility.

Name	*Operation*	*Operands*
[symbol]	RPL	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 , LEVENT=rpl exit routine address[1] , GEVENT=rpl exit routine address[1] }] [, BRANCH={ YES[1] | NQ }] [, OPTCD= { [, { CONALL | CONANY }] [, { ACCEPT | ACQUIRE }] [, { SPEC | ANY }] [, { QUIESCE | STOP[2] | START[2] }] [, { PASS | RELEASE }] [, { LOGONMSG | DEVCHAR | COUNTS | TERMS | APPSTAT | CIDXLATE | TOPLOGON }] [, { SYN | ASY }] [, { CS | CA }] [, { ED | CD }] [, { BLK | LBM | LBT }] [, { CONV | NCONV }] [, { COND | UNCOND | LOCK }] [, { ERASE | EAU | NERASE }] [, { RELRQ | NRELRQ }] [, { Q | NQ }] }]

[1] *The LEVENT, GEVENT, and BRANCH=YES operands are only valid in OS/VS2.*
[2] *The OPTCD=STOP | START operand is only valid in OS/VS1 and OS/VS2.*

symbol

Function: Provides a name for the macro instruction, and thus for the RPL generated by it. This name can be used by any RPL-based macro instruction.

AM=VTAM

Function: Indicates that a VTAM RPL is to be built. This operand is required only for programs running under DOS/VS; the DOS/VS assembler builds a VSAM RPL if this operand is not coded. The AM operand is ignored by the OS/VS1 and OS/VS2 assemblers.

ACB=acb address

Function: Associates the request that will use this RPL with an ACB. All requests that use the RPL must indicate an ACB.

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

NIB=nib address

Function: 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.

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

Note: The NIB field can be set with any of the above macro instructions. For example:

```
CHANGE   RPL=RPL1,NIB=NIB6
```

Although these macro instructions use a NIB address to indicate a terminal, the READ, WRITE, SOLICIT, RESET, and DO macro instructions use a CID to indicate a terminal (and CLSDST works either way). 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 bit set in the first byte. 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 generated 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 rule also applies to the GENCB and MODCB macro instructions.

AREA=data area address

Function: AREA is a multipurpose operand; the use to which it is put depends on the request that is using the RPL.

When used by a SIMLOGON, INTRPRET, or a CLSDST macro with a PASS option code, AREA indicates the address of an area containing a logon message.

When used by a 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.

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

AREALEN=data area length

Function: 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.

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

RECLEN=data length

Function: RECLEN is a dual-purpose operand; the use to which it is put depends on the request that is using the RPL.

When used by a SIMLOGON, INTRPRET, or CLSDST macro with a 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 READ or WRITE macro, RECLEN indicates the length (in bytes) of the data that begins at the address indicated by AREA. For WRITE operations, RECLEN provides the application program a means of telling VTAM how much data is to be transferred. For READ operations, the RECLEN *operand* has no meaning; however the four-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 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.

The application program can obtain the value set 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),              OBTAIN THIS RPL'S...              X
        FIELDS=RECLEN,        ...RECLEN FIELD...                X
        AREA=WORKAREA,        ...AND PLACE IT IN WORKAREA...    X
        LENGTH=4              ...WHICH IS FOUR BYTES LONG.
```

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

AAREA=alternate data area address

Function: AAREA is a multipurpose operand; the use to which it is put depends on the request that is using the RPL.

When used by a CLSDST macro instruction with a PASS option code, AAREA indicates the location of an eight-byte data 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 an input work area where VTAM places an application program identification (eight bytes long), and possibly a translated logon sequence length indicator (one byte long) and a translated logon 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.

AAREALN=alternate data area length

Function: Indicates the length (in bytes) of the data area identified by the AAREA operand. When AAREA is used as an input area for a conversational write operation, VTAM will use 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.

ECB=event control block address

Function: 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.

Note: ECB posting is performed only if asynchronous handling of the connection or I/O request has been specified (ASY option code in the RPL).

The application program must issue a CHECK macro to determine whether posting has occurred, and to determine the status of the I/O event.

If the ECB operand is specified, the EXIT, LEVENT, or GEVENT operands must not be specified. These represent two alternative ways for the application program to be notified when the asynchronous operation is complete. (EXIT-LEVENT-GEVENT indicate a routine to be entered when the operation is completed.)

If neither an ECB *nor* an EXIT routine is indicated, the ECB field in the RPL is used as an ECB, rather than as the address of an ECB. Once ECB or EXIT has been specified, however, this 'internal' ECB is never used.

{EXIT|LEVENT|GEVENT}=rpl exit routine address

Function: Indicates the address of a routine to be scheduled when the request represented by this RPL is completed. The use of the LEVENT or GEVENT keywords indicates to VTAM that the routine is to be executed in supervisor state, as indicated below. LEVENT and GEVENT can only be used by application programs authorized to do so by the installation.

EXIT=rpl exit routine address

The RPL exit routine is executed in problem state.

LEVENT=rpl exit routine address (OS/VS2 Only)

The RPL exit routine is scheduled for execution under a *local* SRB (system request block); as a consequence, the exit routine is executed in supervisor state with a high dispatching priority. All READ, WRITE, and SOLICIT requests issued in this routine must have their RPL's BRANCH field set to YES. **This type of scheduling should be used only by those who are thoroughly familiar with the restrictions that apply to routines scheduled under an SRB.**

GEVENT=rpl exit routine address (OS/VS2 Only)

The effect of GEVENT is the same as the effect of LEVENT, except that the RPL exit routine is scheduled for execution under a *global* SRB. **This type of scheduling should be used only by those who are thoroughly familiar with the restrictions that apply to routines scheduled under an SRB.**

Note: If the SYN option code has been specified, this operand is ignored.

The RPL exit routine is scheduled only if asynchronous handling of the request has been specified. When the routine receives control, the three-byte FDBK (feedback) field in the RPL will be filled in. This field will indicate the status of the request whose completion caused the routine to be invoked. (See Appendix A for information regarding the use of the feedback field.)

The FDBK examination could reveal that the request was completed with an error that would have invoked the SYNAD or LERAD exit list routines had the RPL exit routine not been invoked in their place. Issuing a CHECK macro instruction in the RPL exit routine will schedule the SYNAD or LERAD exit list routines, as appropriate, as well as setting the RPL to an inactive status. (LERAD and SYNAD exits are discussed in the EXLST macro instruction description.)

When the EXIT routine receives control, these general purpose registers indicate the following:

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.

Register 15 -- the address of the RPL exit routine.

If the EXIT, LEVENT, or GEVENT operand is specified, the ECB operand must not be specified. (The EXIT-LEVENT-GEVENT field and the ECB field occupy the same storage area in the RPL.)

BRANCH= { YES |NO }

Function: For LEVENT or GEVENT RPL exit routines, or for asynchronous exit list routines authorized to use the LE or GE attribute (see EXLST macro), BRANCH indicates the action to be taken when a READ, WRITE, or SOLICIT macro instruction is issued.

YES (OS/VS2 Only)
When the macro instruction is executed, a direct branch is taken to the VTAM routines that process the macro instruction. Routines being executed under an SRB must use this operand in order to use the READ, WRITE, or SOLICIT macro instruction. Routines *not* being executed under an SRB must *not* use BRANCH=YES.

NO
When the macro instruction is executed, an SVC interruption will be used to transfer control to the VTAM routines. BRANCH=NO must always be in effect for READ, WRITE, and SOLICIT requests that are issued outside of the special routines mentioned above. For DOS/VS and OS/VS1, all requests are handled as though BRANCH=NO had been specified; if the BRANCH operand is specified in these programs, it is ignored.

OPTCD= { option code | (option code,...) }

Function: 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, omit the parentheses.

```
RPL     ACB=ACB1,OPTCD=(SPEC,SYN,CS)
RPL     ACB=ACB1,OPTCD=SPEC
```

Note: The MODCB macro instruction can be used to change the option codes in effect for the RPL after it has been built.

{ **CONANY** |**CONALL** }
When an OPNDST macro instruction (with an ACQUIRE option) is issued and the NIB field if its RPL indicates a list of NIBs, this option code indicates the following:

CONANY
Connection is to be made to the first available terminal of the NIB list indicated by the NIB field. Control is returned to the application program after this one connection has been made.

CONALL
Connection is to be made to *all* the terminals in the list. The connections are not made until all the terminals are available. Unless the ASY option code is in effect, control is not returned to the application program until all of the terminals have been connected. When this has been accomplished, all the NIBs in the NIB list can be used for I/O requests.

When a SIMLOGON macro instruction is issued and the NIB field of its RPL indicates a list of NIBs, this option code indicates the following:

CONANY
A simulated logon request 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 an active one exists, when this one logon request 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 request was generated.

CONALL
Logon requests are to be generated for *all* the terminals represented in the NIB list. The logon requests are not generated until all the terminals are available. Unless the NQ option code is in effect, control will not be passed to the application program's LOGON exit routine until all the logon requests have been generated.

{ **ACCEPT**|**ACQUIRE** }
Indicates whether OPNDST is being issued to accept a terminal's logon request or whether it is being issued to acquire that terminal.

ACCEPT
VTAM connects the application program to a terminal that has issued a logon request. If more than one terminal has issued a logon request and is waiting to be accepted, the first terminal that issued a logon request is connected. The symbolic name of that terminal is placed in the NIB pointed to by OPNDST's RPL. If the SPEC option 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 request.

ACQUIRE
VTAM connects the application program to the terminal represented by this NIB if the terminal has *not* issued a logon request. If this NIB is the first in a

list of NIBs, the CONALL-CONANY processing option determines which of the terminals represented in the list (that have not issued logon requests) are connected. If CONALL is in effect, all of the terminals represented in the list are connected. If CONANY is in effect instead, only the first available terminal represented in that list is connected.

For programs running under OS/VS1 or OS/VS2, the use of ACQUIRE must be authorized for the application program by the installation.

{ **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 when that terminal issues a logon request to the application program. The terminal is identified by referring to its associated NIB.

ANY
Connection is to be made to any terminal that has issued a logon request for the application program.

When the RPL is used by a READ or SOLICIT 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 is to be obtained from any one terminal currently connected to the application program and not already engaged in a dialog with it.

For SOLICIT, data is to be obtained from all of the terminals currently connected to the application program, subject to the setting of the CS-CA option code.

{ **QUIESCE**|**STOP**|**START**
Indicates how a SETLOGON request is to affect the queuing of logon requests.

QUIESCE
VTAM stops queuing logon requests for the ACB indicated in the ACB field. Logon requests already queued are not affected. OS/SV1 and OS/VS2 programs should use this option only to *permanently* stop logon request queuing.

START
VTAM resumes logon request queuing. This type of SETLOGON request is not used to start logon request queuing when the ACB is opened (this occurs automatically during OPEN processing if the ACB's MACRF field is set to LOGON); it is instead used to start logon request queuing that has been suspended by a SETLOGON request issued with the STOP option.

STOP
VTAM stops queuing logon requests for the ACB. Use this type of SETLOGON request to *temporarily* stop logon request queuing (and use the START option of SETLOGON to resume logon request queuing).

{ **PASS|RELEASE** }

Indicates whether or not a simulated logon request is to be generated when a CLSDST macro instruction is issued.

PASS
VTAM generates a simulated logon request on behalf of the terminal being disconnected and directs these requests 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 will also send a logon message with the logon request. VTAM obtains the message from the storage area identified in the AREA field, and sends the number of bytes indicated in the AREALEN field. For application programs running under OS/VS1 or OS/VS2, the use of CLSDST with PASS must be authorized by the installation.

RELEASE
No simulated logon request is generated; the terminal is simply disconnected from the application program.

{ **LOGONMSG|DEVCHAR| COUNTS|TERMS|APPSTAT|CIDXLATE|TOPLOGON** }

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 request for the application program. A terminals's logon message can be retrieved only once.

The RPL's NIB field must point to a NIB whose NAME field contains the symbolic name of the terminal whose logon message is to be retrieved.

This information is provided in the parameter list passed to the LOGON exit list routine.

The RPL's ACB field must indicate the ACB to which the logon request was directed.

The AREA and AREALEN fields must indicate the location and length of the storage area where the logon message is to be placed. Logon message specifications are determined by the installation during VTAM definition.

DEVCHAR
INQUIRE obtains the device characteristics of a terminal, as they are defined in the resource definition table at the time INQUIRE is executed. These device characteristics can be used to define which processing options the program wants to be in effect for the NIB associated with the terminal.

Either 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. 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 currently connected via a given ACB, and the number of terminals that have requested logon via that ACB but have not yet been connected. These two numbers are placed in an eight-byte area in the 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 four bytes and the number of terminals requesting logon in the second four bytes.

The connection and logon requests 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.

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 either a terminal entry or a group 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 processing options are set according to the underscored (assumed) values shown for the PROC operand in the NIB macro instruction, and 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, and the NIBs are then ready to be used for connection.

APPSTAT
This type of INQUIRE determines whether a given application is available or unavailable. An available application is one whose ACB is active (open) and indicates that logon requests are to be accepted.

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 fullword area pointed to by the AREA field indicates whether the application program is available or not.

CIDXLATE
INQUIRE provides the symbolic name of the terminal whose CID you supply.

The RPL's ARG field must contain the CID of the terminal. The eight-byte symbolic name of that terminal is returned in the data area indicated in the AREA field.

TOPLOGON
For a given ACB, INQUIRE provides the symbolic name of the terminal that is currently at the top of the logon request queue for that ACB (that is, the terminal that has spent the greatest length of time waiting for its logon request to be satisfied).

The RPL's ACB field must indicate the ACB whose logon queue is to be used. The eight-byte symbolic name is returned in the data area indicated in the AREA field.

{**SYN**|**ASY**}

Indicates whether VTAM should synchronously or asychronously handle any connection, logon, or I/O request made via 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. When control is returned to the application program, register 15 will contain a completion code.

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 operand is in effect for the RPL, VTAM posts a completion indicator in the event control block indicated by this operand. (If neither an ECB nor an EXIT address is specified in the RPL, the ECB field itself is used as an event control block.) The application program must issue a CHECK macro to determine whether the ECB has been posted and to determine the status of the operation.

If the EXIT operand is in effect for the RPL, VTAM schedules the exit routine indicated by this operand. This exit routine could issue the CHECK macro to cause automatic entry into a LERAD or SYNAD exit list routine if the requested operation ends with a logical or physical error. CHECK should be issued here if the application program has no active LERAD or SYNAD exit list routine, since CHECK will return a code indicating whether or not a logical a physical error occurred. CHECK should also be issued if the application program is to check for additional types of unusual completion (see the CHECK macro for details).

Note: After an asynchronous request has been accepted, and before that request has been completed, do not modify the RPL, ACB, EXLST, or NIB control blocks associated with the request. 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.

{**CS**|**CA**}

The CS (continue specific) and CA (continue any) option codes apply to I/O requests and indicate whether or not data solicited from a terminal can be obtained with a READ macro instruction having the ANY option code in effect.

CS

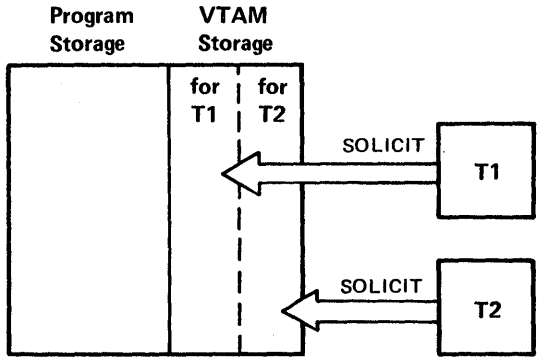
Data solicited from the terminal can be obtained *only* with a READ macro instruction having the SPEC option code in effect. The arrival of data from the terminal does not trigger the completion of a READ macro that was issued with the ANY option code specified.

CA

Data solicited from the terminal can be obtained by either kind of request -- READ with the SPEC option code or READ with the ANY option code.

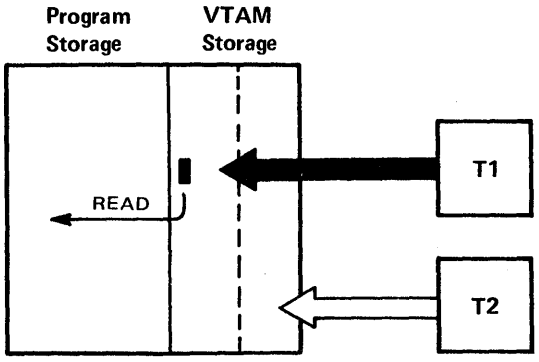
The reason for the CS-CA option code can perhaps be more readily understood by considering the following example. The first four frames illustrate a situation in which failure to use the CS option code could cause unpredictable results. The fifth frame shows this situation remedied with the CS option code.

▶ SOLICIT
▶ SOLICIT

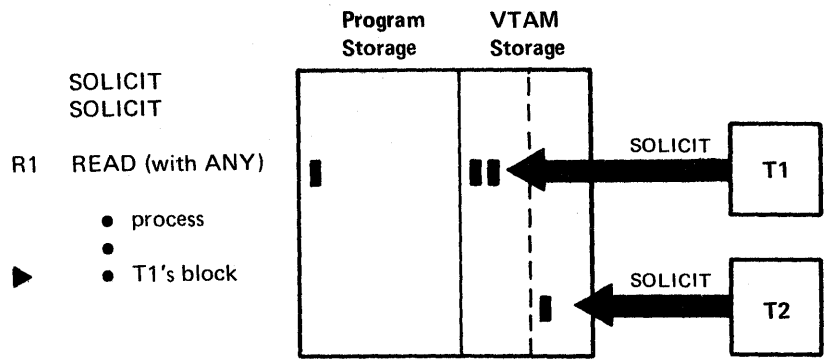


An application program issues SOLICIT macro instructions to obtain data from two terminals connected to it (T1 and T2).

SOLICIT
SOLICIT
▶ R1 READ (with ANY)

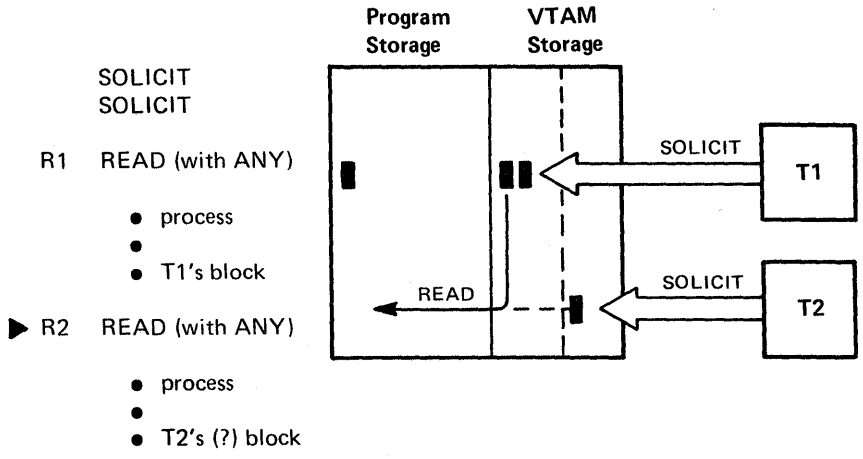


The application program also issues a READ with an ANY option code; this brings into program storage a block of data from the first terminal that responds to the solicitation. In this example, T1 responds first and the pending READ macro instruction is completed. The application program can proceed to process this block of data.

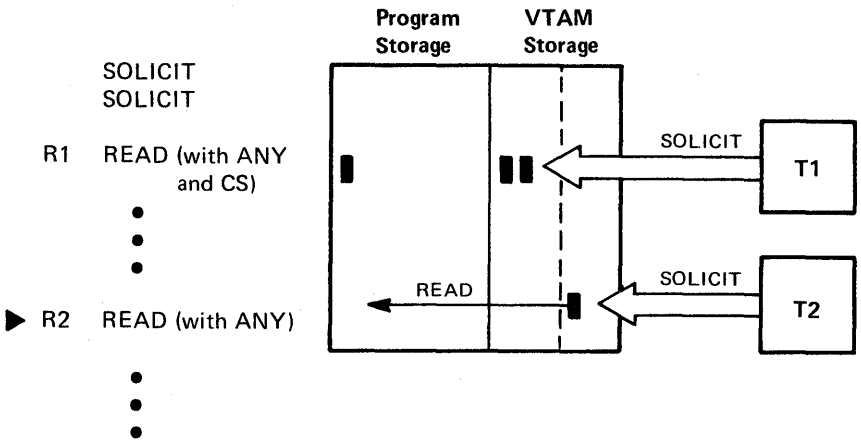


While the application program is still processing T1's first block of data, the on-going solicit operations bring in two more blocks of data from T1 and a block of data from T2. (Whether SOLICIT obtains only one block, or a message or transmission, depends on the setting of the BLOCK-MSG-TRANS-CONT processing option. Here transmissions consisting of several blocks are being solicited.)

If the processing of T1's first block involves any time-consuming operations (like performing I/O with a direct-access storage device), the application program may want to asynchronously handle the arrival of T2's data with another READ with OPTCD=ANY. That is, the application program may want to go on and retrieve T2's data while still working on T1's data.



R2 may not retrieve T2's data however; if T1's second block had arrived before T2's data, the second READ will move T1's second block into program storage. If the application program cannot handle T1's subsequent blocks until it is finished handling T1's first block, the application program will have to wait until the processing of the direct access I/O operation is completed.



This problem can be resolved by the use of the CS-CA option code. When an application program issues an I/O request with the CS option code specified, it is in effect saying "place the terminal that responds to this request in a status wherein only *specific* read requests can obtain data solicited from it." Had this option been used for the first READ macro instruction, only the arrival of data from a terminal *other than T1* could trigger the completion of the second READ macro instruction:

Note: After the first block of data has been received and processed, READ with OPTCD=SPEC macro instructions are used to obtain the remaining blocks. These macro instructions should also have CS set in their RPL. On the last I/O request directed specifically at the terminal – when the terminal is re-solicited with a new SOLICIT macro instruction, for example – the CA option code should be used. This will return the terminal to its original status; data obtained by the new SOLICIT can again trigger any waiting READ macro instructions having the ANY option code.

{**ED**|**CD**}

ED
VTAM will end the dialog.

CD
VTAM will continue the dialog.

When this option code is used with a SOLICIT, READ, WRITE, or DO macro instruction, the application program is requesting that a *dialog* should continue or should end.

A dialog is approximately equivalent to Network Control Program (NCP) session.* A dialog is started when an input or output transmission begins. A dialog will end with the end of the transmission unless CD is specified.

The effect of continuing a dialog is primarily to give scheduling priority to the terminal that is in dialog, at the expense of terminals on the same line that are not in dialog. A disadvantage to a dialog, however, is that additional 3704 or 3705 communications controller storage is required. (Scheduling priority is the priority assigned to a terminal on a multipoint line that determines whether or not an I/O request directed at that terminal will take precedence over a request that is already queued for another terminal on the same line.)

The degree of scheduling priority obtained with CD depends on the number of terminals on the line, and on the NCP session and transmission limits established by the installation (these limits are explained in the publication cited below). CD has no effect on scheduling priority if there is only one terminal on the line. CD likewise has no effect if the NCP session limit is as great as the number of terminals on the line, or if the transmission limit is one. CD gives absolute scheduling priority (that is, holds the line for exclusive use of the terminal) if the session limit is one and there is no transmission limit. As the session limit decreases, or the transmission limit increases, CD is more likely to result in a high scheduling priority in the 3704 or 3705 for the terminal. The negative poll limit for the Network Control Program may also effect the scheduling priority obtained by specifying CD.

In general, the use of CD should be restricted to situations in which the time between two I/O operations is expected to be very short relative to the time required for the I/O operations themselves. This might be the case, for example, when a transmission sent from the terminal at keying speed is to be read and automatically (immediately) followed by a write operation.

To take full advantage of CD, you should be aware of such factors as the duration of the I/O operations, the probable response times, the number of terminals per line, and the NCP session and transmission limits for that line. If you are not aware of these factors, avoid the use of CD.

{**BLK**|**LBM**|**LBT**}
Indicates that the block of data to be transferred on an output operation represents a block and nothing more (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.

{**CONV**|**NCONV**}
Indicates whether or not a WRITE macro instruction is to be handled as a conversational write request.

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.

*NCP sessions are explained in *IBM 3705 Communications Controller Network Control Program, Generation and Utilities Guide and Reference Manual,* GC30-3000.

NCONV
Only the output operation is performed.

{ **COND**|**UNCOND**|**LOCK** }
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. RESET also resets any error lock that has been set for the terminal.

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.

{ **ERASE**|**EAU**|**NERASE** }
Indicates the action to be taken when a WRITE macro instruction is issued.

ERASE
WRITE erases the screen of a display device attached to a 3270 Information Display System or a 2770 Data Communication System, and then send 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.

NERASE
WRITE performs an ordinary write operation, with no display screen erasure.

{ **RELRQ**|**NRELRQ** }
Indicates the action to be taken when (1) an OPNDST macro with ACQUIRE, or a SIMLOGON macro is issued, and (2) the terminal that is the object of this connection or simulated logon request is already connected to another application – that is, already connected to an ACB other than the one being used for the OPNDST or SIMLOGON macro.

RELRQ
If the application to which the terminal is currently connected has an active RELREQ exit in its exit list, the RELREQ exit routine is invoked.

NRELRQ
No RELREQ exit routine is invoked.

{ **Q**|**NQ** }
Indicates the action VTAM is to take when the application program requests connection (with the OPNDST macro) and the terminal that is to be the object of

this request is unavailable. (The terminal is unavailable if it is currently connected to another application program.)

Q
VTAM is to satisfy the request when the terminal is finally available, and notify the application program when it has done so. As is true with most of the RPL-based requests described in this book, the nature of this notification depends on the SYN-ASY option code, and the RPL's ECB and EXIT field contents:

If synchronous request handling has been requested (SYN option code in effect for the RPL), the application program receives control back only after connection has been established or attempted.

If asynchronous request handling and ECB posting has been requested (ASY option code and ECB in effect for the RPL), the ECB is posted when the request has been satisfied. The application program must issue a CHECK macro to determine whether the ECB has been posted.

If asynchronous request handling (ASY option code in effect for the RPL) and an EXIT routine address have been specified, the EXIT routine is scheduled when the request has been satisfied.

NQ
If the terminal is not currently available, VTAM is not to wait until the terminal is available, but is instead to immediately return control to the application program. (Without the NQ option, a connection or simulated logon request might remain pending indefinitely, waiting for another application program to release the terminal.)

Note: When NQ is specified and control is finally returned to the application program, the request may or may not have been satisfied. Register 15 (and also the first byte of the RPL's FDBK field) is set to hexadecimal 10 to indicate that this has happened.

Examples

```
RPL1     RPL   ACB=ACB1,NIB=NIB1                          X
                OPTCD=(SPEC,ASY),                          X
                EXIT=EXITPGM,AM=VTAM
```

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 completes, the application program will be interrupted, and the routine at EXITPGM is invoked.

```
RPL2     RPL   ACB=ACB1,ARG=(6),                          X
                AREA=SOURCE,RECLEN=132,                    X
                ECB=ECBWORD,OPTCD=ASY,AM=VTAM
```

RPL2 can be used by a WRITE macro instruction to write a block of data (132 bytes from SOURCE) to the terminal whose CID is in register 6. When the request has been scheduled, control is returned immediately. When the request has been completed, ECBWORD is posted. (The CHECK macro instruction used to check the write operation would point to RPL2.)

RPL Fields and RPL -- Oriented Macro Instructions

The following figure 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 these macro instructions -- either 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.

RPL-modifying macro instructions ⇒ / Applicable RPL fields:	(ACQUIRE) OPNDST	(ACCEPT) OPNDST	(PASS) CLSDST	(RELEASE) CLSDST	CHANGE	SIMLOGON	SETLOGON	SOLICIT	READ	WRITE	RESET	INQUIRE	INTRPRET	DO
ACB	A	A	A	A	A	A	A	A	A	A	A	A	A	A
ARG/	V	V	A	A				A	AV	A	A			A
NIB	A	A	A	A	A	A						A	A	
AREA	V		A			A			A	A		A	A	A
AREALEN									A			A		
RECLEN			A			A			V	A		V	A	V
AAREA			A							A			A	V
AAREALN										A			A	
ARECLEN										V			V	
BRANCH								A	A	A				
ECB/EXIT	A	A	A	A	A	A	A	A	A	A	A	A	A	A
FDBK	V	V	V	V	V	V	V	V	V	V	V	V	V	V
USER	V	V						V	V	V	V	V		V
OPTCD:														
SPEC-ANY		A						A	A					
QUIESCE-START-STOP							A							
PASS-RELEASE			A	A										
LOGONMSG-DEVCHAR-COUNTS-TERMS-APPSTAT-CIDXLATE-TOPLOGON												A		
SYN-ASY	A	A	A	A	A	A	A	A	A	A	A	A	A	A
CS-CA	A	A		A				A	A	A	A			A
ED-CD								A	A	A				A
BLK-LBM-LBT										A				
CONV-NCONV										A				
COND-INCOND-LOCK											A			
ERASE-EAU-NERASE										A				

The presence of a symbol means that the RPL field or option code is applicable for the macro instruction in one of these two ways:

[A] The field or option code is set by the **application program** to supply VTAM information about the request.

[V] The field is set by **VTAM** when the request has been processed.

Figure 5. RPL Fields Applicable to the Macro Instructions that can Modify RPLs

SETLOGON -- Reset an ACB's Logon Status

When an application program opens an ACB that has MACRF=LOGON specified for it, VTAM begins queuing logon requests that are directed at the ACB. SETLOGON provides a means of preventing VTAM from queuing logon requests.

There are three types of SETLOGON request: QUIESCE START and STOP. An RPL option code determines which is used when a SETLOGON macro instruction is executed.

The QUIESCE version of SETLOGON (and the only version available to programs running under DOS/VS) causes VTAM to permanently prevent logon request queuing. An application program might want to use this type of SETLOGON prior to issuing a CLOSE macro instruction.

Applications running under OS/VS1 and OS/VS2 have two extra options of SETLOGON available to them. These allow the application program to *temporarily* stop logon request queuing (STOP option code) and later resume logon request queuing (by issuing SETLOGON with the START option code).

Since logon requests might accumulate much faster than the application program can handle them, the application program can use the STOP option of SETLOGON to temporarily place itself in an 'unavailable" status. (The INQUIRE macro instruction with the COUNTS option code can be used to determine how many logon requests are queued.)

To summarize:

OPEN	ACB's MACRF field set to LOGON.	Logon request queue opened.
SETLOGON	RPL=RPL1, OPTCD=STOP	(Available under OS/VS1 and OS/VS2 only.) Logon request queue *temporarily* closed.
SETLOGON	RPL=RPL1, OPTCD=START	(Available under OS/VS1 and OS/VS2 only.) Logon request queue reopened.
SETLOGON	RPL=RPL1, OPTCD=QUIESCE	Logon request queue closed.

Note: SETLOGON can be used only if the ACB is opened with MACRF=LOGON. If the ACB is opened with MACRF=NLOGON, the only way an application program can reset the ACB's logon status is to close the ACB, reset its MACRF field to LOGON, and then reopen.

Name	*Operation*	*Operands*
[symbol]	**SETLOGON**	**RPL=rpl address** **[, rpl keyword=new value]...**

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL whose ACB field points to the ACB for which logon requests are to be stopped or allowed.

rpl keyword=new value

Function: Indicates an RPL field to be modified, and the new value that is to be contained or represented within it.

Format: For *rpl keyword* 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.

Note: See Figure 5 at the end of the RPL macro instruction description for a list of all the RPL fields that might be modified here.

Example

```
NOMORE      SETLOGON    RPL=RPL1,ACB=ACB1,OPTCD=STOP
                              .
                              .
                              .
RESUME      SETLOGON    RPL=RPL1,ACB=ACB1,OPTCD=START
```

NOMORE causes VTAM to stop queuing logon requests directed to ACB1. Any logon requests directed to ACB1 after NOMORE has been executed will be rejected. When RESUME is executed, VTAM will again queue logon requests for ACB1 as they occur. This example assumes that the MACRF field of ACB1 has been set to LOGON. Note that this example applies only to programs running under OS/VS1 or OS/VS2.

Return of Status Information

After SETLOGON processing is finished, register 15 indicates one of the following hexadecimal values:

0 If the ASY option code is in effect, VTAM accepted the SETLOGON request. If the SYN option code is in effect, queuing of logon requests has been successfully stopped or started for the ACB.

4 The request cannot be accepted because the RPL is being used by another request. The RPL's FDBK field has not been set. If an active LERAD exit list routine is available, it has been invoked.

8 A logical error (see Appendix A) occurred; the FDBK field can be examined to determine which one it was. If an active LERAD exit list routine is available, it has been invoked. (This return code is possible only if the SYN option code is in effect.)

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.

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 always set by VTAM can also be displayed with SHOWCB. All of the fields applicable for SHOWCB are shown in Figure 6 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 returns an error code and places the required length indicated in register 0. Figure 6 shows the required lengths for all the control block fields that can be displayed with SHOWCB.

List 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={ L | (E, parameter list address) }]**

symbol

Function: Provides a name for the macro instruction.

AM=VTAM

Function: Identifies this macro instruction as a VTAM macro instruction. This operand is required for the DOS/VS assembler, but is ignored by the OS/VS1 and OS/VS2 assemblers.

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

Function: Indicates the type and location of the control block whose fields are to be extracted.

Format: Code only one of the operands. For the list form of SHOWCB, register notation cannot be used.

Note: One of these operands must be selected for the standard or list forms; but for the execute form, it is optional.

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

Function: Indicates the control block field or fields whose contents are to be extracted.

Format: For *field name* code one of the field names that appear in the first column of Figure 6 below. 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).

Note: This operand is required for the standard and lists forms of SHOWCB, but for the execute form, it must not be specified.

AREA=work area address

Function: 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.

Format: Code the address of a work area that begins on a fullword boundary. For the list form of SHOWCB, register notation cannot be used.

Note: This operand is required for the standard and list forms of SHOWCB, but for the execute form it is optional.

LENGTH=work area length

Function: Indicates the length (in bytes) of the storage area designated by the AREA operand.

If this length is insufficient, SHOWCB returns a hexadecimal 10 in register 15 and the required length in register 0. The required length for each field are shown in the second column of Figure 6.

Format: Code the number of bytes in AREA either as a numerical value or as an expression that will be equated to a numerical value. Register notation is not permitted in the list form of SHOWCB.

Note: This operand is required for the standard and list forms of SHOWCB, but for the execute form it must not be specified.

MF= { L | (E, parameter list address) }

Function: Indicates that either a list form or an execute form of SHOWCB is to be used.

MF=L

The list form (L-form) of this macro instruction creates a parameter list for later use by the by the execute form. Because the L-form macro instruction generates only this parameter list, and no executable code, operand forms like register notation are prohibited. Only relocatable expressions valid for adcons can be used. The user is responsible for branching around the generated parameter list, which is variable in length.

MF=(E,parameter list address)
The execute form (E-form) of this macro instruction can modify the parameter list generated by its list form. The expansion of the execute form provides the executable instructions required to perform parameter list modification and passing of control. The *parameter list address* should specify the location of the list form of the macro instruction. Register notation can be used.

Note: Although the execute form of SHOWCB can modify the list form's parameter list, it cannot add to it. Therefore if an operand value will not be known until program execution and is to be supplied with the execute form, the list form must also specify that operand and supply a dummy value for it.

For example: Assume that the contents of an ACB's APPLID and EXLST fields are to be placed in an area whose address will become available in register 3 during program execution. First code a list form of SHOWCB as follows, taking care to specify the AREA operand and supply some value for it --

```
LFORMS    SHOWCB    ACB=ADCON1,AREA=0,LENGTH=8                X
                    FIELDS=(APPLID,EXLST),MF=L,AM=VTAM
```

and code an execute form like this:

```
EFORMS    SHOWCB    AREA=(3),MF=(E,LFORMS)
```

Examples

```
SHOW1     SHOWCB      NIB=NIB1,FIELDS=NAME,                   X
                      AREA=NAME1,LENGTH=4,AM=VTAM
```

SHOW1 extracts the contents of NIB1's NAME field and places it in NAME1.

```
SHOW2     SHOWCB    RPL=RPL1,                                 X
                    FIELDS=(FDBK,ARG,AREA,RECLEN)             X
                    AREA=(3),LENGTH=16,AM=VTAM
```

SHOW2 extracts the contents of RPL1's FDBK, ARG, AREA, and RECLEN fields and place 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 these general purpose registers to indicate the following:

Register 0: If the length of AREA is too small, this register contains the correct length needed to hold the fields.

Register 15: The hexadecimal content of this register indicates completion status:

0 The contents of all of the fields were successfully moved.

4 The indicated control block is not of the type it should be. If, for example, ACB=MYACB were specified, this return code means that the control block found at MYACB is not a valid ACB.

8 The execute form was used incorrectly. For example, a control block field is specified in the execute form instead of in the list form.

C Required information was not supplied. For example, a control block location was indicated, but no field in that control block was specified.

10 The length of the storage area where the field contents were to have been placed is too small. Check register 0 for the correct length.

The field names shown in the first column 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 codes; see OPEN macro instruction

EXLST Fields

Field Name	*Length (bytes)*	*Description*
LERAD	4	Address of LERAD exit list routine
SYNAD	4	Address of SYNAD exit list routine
UNSIP	4	Address of UNSIP exit list routine
ASYIP	4	Address of ASYIP exit list routine
TPEND	4	Address of TPEND exit list routine
RELREQ	4	Address of RELREQ exit list routine
LOGON	4	Address of LOGON exit list routine
LOSTERM	4	Address of LOSTERM exit list routine
ATTN	4	Address of ATTN exit list routine
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 area or logon message
AREALEN	4	Length of AREA, in bytes
RECLEN	4	Length of data in 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	Fullword for posting connection or I/O completion information, or address of same
EXIT	4	Address of RPL exit routine
LEVENT	4	Address of RPL exit routine
GEVENT	4	Address of RPL exit routine
FDBK	4	Three-byte feedback field, right-justified
USER	4	Arbitrary data originally set in a NIB's USERFLD field
RPLLEN	4	Length of RPL, in bytes

NIB Fields

Field Name	*Length (bytes)*	*Description*
NAME	8	Symbolic name of terminal
USERFLD	4	Arbitrary data associated with NAME
CID	4	Communication ID – shortened form of NAME
NIBLEN	4	Length of NIB, in bytes
DEVCHAR	8	Device characteristics

Figure 6. Control Block Fields Applicable for SHOWCB

SIMLOGON -- Generate a Simulated Logon Request

A logon request is a request for connnection made by a terminal and directed towards an application. If the application program has an active LOGON exit routine, it is invoked. There, presumably, VTAM is asked to connect the terminal to the application program; that is, an OPNDST macro is issued.

A logon request can be initiated by the terminal itself, or an application program can initate the logon request on *behalf* of the terminal. The latter is called a *simulated logon request,* and the program uses the SIMLOGON macro instruction to generate it.

By issuing SIMLOGON, the application program can use its LOGON exit list routine to service *self-initiated* logon requests. When used this way, SIMLOGON is equivalent to an OPNDST connection request with an ACQUIRE processing option, except that the LOGON exit list routine handles the connection request.

Note: Do not issue SIMLOGON if no active LOGON exit list routine will be available when SIMLOGON is executed.

The SIMLOGON macro indirectly indicates the terminal on behalf of which the logon request is to be made. The RPL operand of the SIMLOGON macro instruction must indicate the address of an RPL whose NIB field contains the address of a NIB. A logon request is generated, at a minimum, for the terminal represented by this NIB. If the NIB field points to a single NIB (one whose LISTEND field is set to YES), one logon request is generated for that terminal. However if the NIB field points to a NIB *list* (LISTEND field set to NO), the effect of SIMLOGON depends on the setting of the CONANY-CONALL processing option in the NIB. These effects are described in detail in the processing option descriptions of the NIB macro, but for convenience they are summarized here as well:

CONANY
A logon request is generated for the first terminal to be released by the application program to which it is currently connected.

CONALL
A series of logon requests are generated, one for each terminal represented in the NIB list. The logon requests are generated until all the terminals are available (that is, are released by any other application programs to which they might be connected).

The RPL must also indicate in its ACB field the ACB to which the simulated logon request is to be generated, and it must indicate in its NIB field the NIB of the terminal for which the request is to be generated.

The SIMLOGON macro instruction can optionally be used to send a logon message along with the logon request. If this is to be done, SIMLOGON's RPL must contain the address of this message in its AREA field, and it must contain the length of this message in its RECLEN field. If a logon message is *not* to be sent, these fields must be set to 0.

For programs running under OS/VS1 or OS/VS2, the use of SIMLOGON must be authorized for the application program by the installation.

Name	*Operation*	*Operands*
[symbol]	SIMLOGON	RPL=rpl address [, rpl keyword=new value]...

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL to be used during SIMLOGON processing. 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 logon requests. The ACB field of the RPL must contain the address of the ACB to which the simulated logon request is to be directed.

rpl keyword=new value

Function: Indicates an RPL field to be modified and the new value that is to be contained within it.

Format: For *rpl keyword* 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.

Note: All of the RPL fields that have a unique effect on SIMLOGON (and thus might be modified here) are discussed above; check Figure 5 in the RPL macro instruction description for a list of all the RPL fields that are applicable for the SIMLOGON macro instruction.

Example

```
SIMLOGON    RPL=RPL1,      RPL MODIFIERS FOLLOW:        X
            ACB=ACB1,NIB=NIBLIST1,                      X
            AREA=LGNMSG
```

This macro instruction generates simulated logon requests for ACB1 from all of the terminals represented in NIBLIST1. Each request will be accompanied by a 60-byte logon message taken from LGNMSG.

Return of Status Information

When the SIMLOGON operation is completed, register 15 indicates one of the following hexadecimal values:

0 If the ASY option code is in effect, the SIMLOGON request was accepted. If the SYN option code is in effect, the simulated logon requests were successfully generated.

4 The request cannot be accepted because the RPL is being used by another request. The RPL's FDBK field has not been set. If an active LERAD exit routine exists, it has been invoked.

8 A logical error (see Appendix A) occurred; the FDBK field of the RPL can be checked to determine which one it was. If an active LERAD exit list routine is available, it has been invoked. (This code can be returned only when the SYN option code is in effect.)

SOLICIT -- Obtain Data from a Terminal

The SOLICIT macro instruction obtains data from one or more connected terminals and places it into VTAM buffer areas. A subsequent READ macro instruction is required to move the data into a program storage area.

SOLICIT performs the preparation or polling required to obtain the data and supplies appropriate line-control responses as blocks of data are obtained. (To solicit data from a 3270 display unit, a command to unlock the keyboard must first be written to the device by the application program.)

As soon as VTAM has accepted the solicit request, control is returned to the program. The operation is then finished as far as the program is concerned; only through the ASYIP exit list routine can the program be notified when data arrives. However, the solicitation of data will continue as long as the terminal's NIB indicates – which could be indefinitely.

Any I/O errors that occur during solicit operations for a given terminal will become known to the program only when it next issues a READ macro for that terminal.

The user of SOLICIT codes the address of an RPL, and (optionally) indicates changed values for RPL fields. Fields in the RPL, and also in the NIB that was used when the terminal was connected, govern the solicit operation. The RPL fields can be modified by the SOLICIT macro instruction itself. The following list shows the effect of the more important of these fields. See the RPL macro instruction description for a list and explanation of all RPL fields applicable for SOLICIT.

The PROC field of the NIB: (*Note*–This field cannot be modified with the SOLICIT macro instruction; use the MODCB and CHANGE macro instructions to do this.)

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. A line-control response is sent to acknowledge receipt of the data obtained from the *previous* solicit, but no such response is sent when data is obtained as a result of *this* SOLICIT macro. The data obtained by this solicit request is acknowledged only when the next solicit request is issued. For programs running under OS/VS1 or OS/VS2, the use of BLOCK must be authorized by the installation.

The effect of BLOCK (and MSG, TRANS and CONT as well) is illustrated in Figure 3 in the NIB macro instruction description.

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 recognized. In effect, this means that data is solicited from the terminal until an entire message has been received. Line-control responses will be sent as each block is received, except for the last block. Its receipt is not acknowledged until the next solicit request is issued.

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. Line-control responses are sent as each block is received, including the last block. Polling does resume until the next solicit request is issued.

PROC=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 request is canceled with the RESET macro instruction, or the terminal becomes disconnected from the program.

The ARG and OPTCD fields of the RPL:

ARG=(register)
If a specific terminal is to be solicited, the ARG field of the RPL must contain the CID for that terminal. ARG=*(register)* is indicated here because that is the only way to place the CID in the RPL with the SOLICIT macro instruction. The CID must first be loaded into a register, and that register indicated with the ARG operand of this macro instruction or in the MODCB macro instruction.

ARG does not apply to SOLICIT when the ANY option code is in effect.

OPTCD={SYN|ASY}
When the SYN option code is in effect, ECB posting does not occur before control is returned to the program, nor is any RPL exit routine invoked. When the ASY option code is in effect, either the ECB is posted or the RPL exit routine (if there is one) is invoked.

OPTCD={CA|CS}
When the CA option code is in effect, the data obtained from the solicit operation is available for a subsequent READ with an ANY option code in effect. When CS is used instead, and the SPEC option code is also in effect, only a subsequent READ with a SPEC option code can be used to retreive the data obtained by the solicit operation. (If the ANY option code is in effect, the CA- CS option code is treated as though CA had been specified – regardless of whether CA or CS is specified.)

OPTCD={SPEC|ANY}
When the SPEC option code is in effect, 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 the ANY option code is in effect, data is solicited from all terminals that are connected to the program but are not engaged in any actual or pending I/O operation with the program.

Name	*Operation*	*Operands*
[symbol]	**SOLICIT**	**RPL=rpl address** **[, rpl keyword=new value]...**

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL that governs the solicit operation.

rpl keyword=new value

Function: Indicates an RPL field to be modified and the new value that is to be contained or represented within it.

Format: For *rpl keyword* 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.

Note: All of the RPL fields that have a unique effect on SOLICIT (and thus might be modified here) are discussed above; check Figure 5 in the RPL macro instruction description for a list of all RPL fields that apply to the SOLICIT macro instruction.

Example

```
SLCT1    SOLICIT    RPL=RPL1,OPTCD=ANY
SLCT2    SOLICIT    RPL=RPL2,OPTCD=SPEC
```

SLCT1 causes data to be solicited from any terminal that has been connected through the ACB indicated in RPL1 and is not currently engaged in communication with the program. 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 solicit request – not when the actual I/O activity is eventually completed. After control has been either returned directly to the next sequential instruction or first passed to an exit routine, these sources of status information may be checked.

The FDBK field of the RPL: Unless the value 4 was returned in register 15, the FDBK (feedback) field indicates the nature of the error. The FDBK field is described in Appendix A.

The USER field of the RPL: When a NIB is established, the user has the option of specifying any arbitrary value in the USERFLD field of that NIB. When a SOLICIT macro instruction with an option code of SPEC in effect is subsequently issued for the terminal represented by that NIB, VTAM obtains the content of the USERFLD field and places it in the RPL's USER field.

Register 15: One of the following hexadecimal values is indicated:

0 VTAM accepted the solicit request. This means that VTAM has checked the control blocks associated with the request and found no information that is contradictory or would otherwise guarantee the eventual failure of the solicit operation. If errors do occur, an error return code will be passed to the program in register 15 when it eventually attempts to read from the terminal involved with the error.

4 The SOLICIT request cannot be accepted because the RPL is being used by another request, or the terminal is not connected to your application program. The RPL's FDBK field has not been set. The LERAD exit routine, if an active one is available, has been invoked.

8 A logical error (see Appendix A) occurred; the FDBK field of the RPL can be examined to determine which one it was. (This return code is possible only when the SYN option code is in effect.)

TESTCB -- Test the Contents of a Control Block Field

TESTCB compares the contents of a specified ACB, RPL, EXLST, or NIB field to 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.

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.

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).

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 controlled 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.

List 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={ L | (E, parameter list address) }]

symbol

Function: Provides a name for the macro instruction.

AM=VTAM

Function: Identifies this macro instruction as a VTAM macro instruction. This operand is required for the DOS/VS assembler, but is ignored by the OS/VS1 and OS/VS2 assemblers.

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

Function: Indicates the type and location of the control block whose field is to be tested.

Format: Use only one of the operands and code the address of the control block. Register notation cannot be used in the list form.

Note: One of these operands must be selected in the standard and list forms, but for the execute form none need be specified.

field name=test value

Function: Indicates a control block field and a value that its contents are to be tested against.

Note: This operand is required for the standard and list forms, but not for the execute form.

Format: For *field name* code any one of the field names listed below.

Group 1: Each field name in this group corresponds to an operand of the macro instruction that generates the field. See the operand descriptions in the ACB, EXLST, RPL, and NIB macro instructions for explanations of each field.

ACB Field Names

AM	EXLST
APPLID	MACRF
PASSWD	

EXLST Field Names

LERAD	ASYIP	LOGON
SYNAD	TPEND	LOSTERM
UNSIP	RELREQ	ATTN

RPL Field Names

ACB	RECLEN	ECB	USER
NIB	ARECLEN	EXIT	BRANCH
AREA	AAREA	LEVENT	OPTCD
AREALEN	AAREALN	GEVENT	

NIB Field Names

NAME	LISTEND
USERFLD	PROC

Group 2: Any of the following field names in this group can also be coded. These names represent fields whose contents are not established by the application program but are instead set by VTAM.

Field Name	*Field Length*	*Description*
ACBLEN	1 fullword	An ACB field containing the length of the ACB, in bytes.
EXLLEN	1 fullword	An EXLST field containing the length of the exit list, in bytes.
RPLLEN	1 fullword	An RPL field containing the length of the RPL, in bytes.

NIBLEN	1 fullword	A NIB field containing the length of the NIB, in bytes.
OFLAGS	N/A	An ACB field that indicates whether or not the ACB is open. This operand is coded as OFLAGS=OPEN. An "equal" condition code indicates that the ACB is activated; an "unequal" condition code indicates that it is not.
IO	N/A	An RPL field that indicates whether the RPL is active or not. This operand is coded as IO=COMPLETE. An "equal" PSW condition code indicates that the RPL is inactive; an "unequal" condition code indicates that the RPL is active.
CON	N/A	A NIB field that indicates whether the terminal represented by the NIB is currently connected to the application program. This operand is coded as CON=YES. An "equal" PSW condition code indicates that the terminal is connected; an "unequal" condition code indicates that it is not.

The rules for coding *test value* depend on what is coded for *field name.*

If *field name* is one of the names listed above in Group 1, code any value that can be supplied with that operand when it is used in an ACB, EXLST, RPL, or NIB macro instruction. The individual operand descriptions for each of these macros indicates how each operand can be specified.

Examples:

```
TESTCB    ACB=ACB1,PASSWD=PSWDAREA,AM=VTAM
TESTCB    EXLST=EXLST1,SYNAD=SYNADPGM,AM=VTAM
TESTCB    RPL=RPL1,AREALEN=64,AM=VTAM
TESTCB    NIB=NIB1,LISTEND=YES,AM=VTAM
```

To test a field listed above under Group 2, code a *test value* that is consistent with the description of that field.

Examples:

```
TESTCB    ACB=ACB1,OFLAGS=OPEN,AM=VTAM
TESTCB    RPL=RPL1,RPLLEN=38,AM=VTAM
```

The A or N (active or inactive) attributes of an exit list (EXLST) entry can be tested as can the address in the exit list entry. The first example below shows how to test for the *presence* of a LOGOFF entry in the exit list, the second example shows how to test for the *active attribute* of the LOGOFF entry, and the third example tests the *address* as well as the active attribute of the entry (an "equal" condition code for the last test would indicate that both the specified address and attribute are correct).

Examples:

```
TESTCB    AM=VTAM,EXLST=EXLST1,LOSTERM=( )
TESTCB    AM=VTAM,EXLST=EXLST1,LOSTERM=(,A)
TESTCB    AM=VTAM,EXLST=EXLST1,LOSTERM=(LSTPGM,A)
```

RPL option codes or NIB processing options (including combinations of them) can also be tested. The test results in an "equal" completion 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    AM=VTAM,RPL=RPL1,OPTCD=(SPEC,CS,BLK)
TESTCB    AM=VTAM,NIB=NIB1,PROC=(MSG,CONFTXT,MONITOR)
```

ERET=error exit routine address

Function: 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 will indicate the nature of the error. These return codes are described at the end of this macro instruction description.

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.

Format: For the list form of TESTCB, register notation cannot be used.

MF={ L|(E,parameter list address)}

Function: Indicates that either a list form or an execute form of TESTCB is to be used.

MF=L

The list form (L-form) of this macro instruction creates a parameter list for later use by the execute form. Because the L-form macro instruction generates only this parameter list, and no executable code, operand forms like register notation are prohibited. Only relocatable expressions valid for adcons can be used. The user is responsible for branching around the generated parameter list, which is variable in length.

MF=(E,parameter list address)

The execute form (E-form) of this macro instruction can modify the parameter list generated by its list form. The expansion of the execute form provides the executable instructions required to perform parameter list modification and passing of control. The *parameter list address should* specify the location of the list form of the macro instruction. Register notation can be used.

Note: Although the execute form of TESTCB can modify the list form's parameter list, it cannot add to it. Therefore if an operand value will not be known until program execution and is to be supplied with the execute form, the list form must also specify that operand and supply a dummy value for it.

For example: Assume that the contents of a RPLLEN field of a particular RPL is to be tested, and the value against which it is to be tested will become available in register 3 during program execution. First code a list form, taking care to code the RPLLEN operand and supply some value for it:

```
LFORMT    TESTCB    AM=VTAM,RPL=ADCON1,RPLLEN=0,MF=L
```

and code an execute form like this:

```
EFORMT    TESTCB    RPLLEN=(3),MF=(E,LFORMT)
```

Examples

```
TEST1     TESTCB    AM=VTAM,ACB=ACB1,MACRF=LOGON
TEST2     TESTCB    AM=VTAM,NIB=NIB1,MODE=BASIC
```

TEST1 compares the contents of ACB1's MACRF field against the code VTAM uses in that field to indicate LOGON. TEST2 compares the contents of the MODE field of NIB1 against the code VTAM uses in that field to represent BASIC. In both cases, the PSW condition code will indicate an equal comparison if the fields do in fact contain the specified value (that is, LOGON and BASIC).

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 one of the following hexadecimal values.

- 0 TESTCB processing was completed successfully; the PSW condition code indicates the result of the test.
- 4 The control block found at the indicated location is not valid. If, for example, RPL=RPL1 was specified, the control block found at RPL1 is not a valid RPL. The test has not been performed, and the PSW condition code is meaningless.
- 8 The execute form was used incorrectly. For example, the execute form specified the control block field and a value to test it with, but the list form did not indicate that field also and supply a dummy test value for it.
- C Required information was not supplied. For example, a control block location was indicated, but the field to be tested within that block was omitted.

WRITE -- Write a Block of Data from Program Storage to a Terminal

The WRITE macro instruction obtains a block of data from a designated area in program storage and sends it to a specific terminal.

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 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).

Should a write operation be pending (or in progress) when another WRITE macro instruction is issued, the first operation is allowed to complete before the second operation is performed. If data is being solicited from a terminal when the WRITE macro instruction is issued, but no data has yet been sent by that terminal, the solicit operation is temporarily interrupted so that the write operation can go ahead. If data transfer is in progress, however, the solicit operation is allowed to complete before the write operation is performed. See the SOLICIT macro instruction for a description of what constitutes and controls the completion of a solicit operation.

If continuous solicitation (PROC=CONT) is specified for OPNDST and a READ or SOLICIT macro instruction is later issued, a WRITE macro instruction cannot be issued until the reading or solicitation is cancelled with a RESET macro instruction.

The user of the WRITE macro instruction codes the address of an RPL. The RPL fields indicate the terminal to be written to, the location of the data that is to be sent to it, and what specific type of write operation is to take place. The RPL fields can be modified with the WRITE macro instruction itself. The following list shows the effect of the more significant of these fields. See the RPL macro instruction description for a list and explanation of all RPL fields applicable for WRITE. The RPL modifiers shown below have the following effects:

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 for an explanation of the CID). ARG=(*register*) is indicated here because register notation must be used if the CID is to be placed in the ARG field with this WRITE macro instruction. The CID can be extracted from the NIB with the SHOWCB macro and then loaded into a register.

AREA=address
The data contained at the location indicated by AREA is sent to the terminal.

RECLEN=length
The number of bytes of data indicated in the RECLEN field is sent to the terminal.

AAREA=address
When the CONV option code is in effect, the data *obtained from* the terminal following the write operation is placed in the storage area indicated by the AAREA field.

AAREALN=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 processing option is taken.

OPTCD= {BLK | LBM | LBT}
These option codes determine whether the line-control characters selected by the system for transmission with the data are for marking 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 in effect, an EOB character (for start- stop devices) or an ETB character (for binary synchronous devices) is sent with the block of data.

When the LBM option code is in effect, an ETX character is sent with the block of data.

When the LBT option code is in effect, an EOB character (for start-stop devices) or an ETX character (for binary synchronous devices) is sent with the block of data (same as LBM). After the block of data is acknowledged by the terminal, an EOT character is sent. The write operation is considered complete as soon as the EOT character has been sent.

OPTCD= {CONV | NCONV}
When NCONV is in effect, no conversational writing is performed. 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 acknowledgment and not data, the action then taken depends of whether the LBM or LBT option code is in effect. For LBM, register 15 and the FDBK field of the RPL are set indicating that no data was received; the write operation is then considered complete. For LBT however, the write operation is not completed until data is eventually received.

When a WRITE with OPTCD=CONV is issued, a second WRITE may not be issued to the same terminal until the first write operation is completed, or unless the first operation is canceled with the RESET macro instruction.

OPTCD= {ERASE | EAU | NERASE}
The ERASE and EAU option codes indicate that one of two special variations of WRITE are to take place; NERASE simply indicates that they are not.

When the ERASE option code is used, the entire display screen of a 2770 Data Communication Terminal or 3270 display device is erased before the block of data is written to the terminal.

EAU means that the unprotected portion of a 3270 display screen is to be erased, and its keyboard unlocked. No data is sent to the terminal.

The following option determines how excess input data is to be handled for a conversational operation (note that this is a NIB processing option, not an RPL option code):

PROC= { TRUNC| KEEP }
When the TRUNC and the CONV option code is in effect, and the incoming data is too big to fit in the area indicated by AAREA, the data is truncated, the remainder is lost, and the write operation terminates with an I/O error.

With KEEP, however, the remainder is saved and passed to the program when the next read request (or conversational write request) is issued for the same terminal.

Name	*Operation*	*Operands*
[symbol]	**WRITE**	**RPL=rpl address** **[, rpl keyword=new value]...**

symbol

Function: Provides a name for the macro instruction.

RPL=rpl address

Function: Indicates the location of the RPL that governs the write operation.

rpl keyword=new value

Function: Indicates an RPL field to be modified and the new value that is to be contained or represented within it.

Format: For *rpl keyword* 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.

Note: All of the RPL fields that have a unique effect on WRITE (and thus might be modified here) are discussed above; check Figure 5. in the RPL macro instruction description for a list of all RPL fields that apply to the WRITE macro instruction.

Examples

```
WRITE1    WRITE    RPL=RPL1,AREA=SOURCE,RECLEN=60,          X
                   EXIT=WRTDONE,OPTCD=ASY
```

WRITE1 sends a 60-byte block of data from SOURCE to a 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 write operation has been scheduled. When the operation is completed, the program is interrupted and control passed to WRTDONE.

WRITE2 erases the unprotected part of a 3270 display screen. The SHOWCB and load instructions are used here to extract that device's CID from NIB1 (assuming that NIB1 was the NIB used when the device was connected) and load it into register 3, where WRITE2 can get it and place it in RPL2.

```
WRITE3    WRITE    RPL=RPL3,AREA=OUTGOING,RECLEN=120        X
                   AAREA=INCOMING,AAREALN=132,
                   ARG=(3),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 in INCOMING. If more than 132 bytes are received, the excess will be lost. Because of the LBT option code, the operation is not completed unless the terminal responds with data.

Return of Status Information

After the WRITE operation is completed, these sources of status information may be checked:

The ARECLEN field of the RPL: If the CONV option code is in effect, ARECLEN indicates the number of bytes of data obtained during the input part of the conversational write operation.

The USER field of the RPL: When a NIB is established, the user can specify any arbitrary value in the USERFLD field of that NIB. When the WRITE macro instruction is subsequently issued for the terminal associated with that NIB, whatever had been placed in USERFLD is placed in the USER field of the RPL by VTAM.

The FDBK field of the RPL: Unless a hexadecimal value of 4 is returned in register 15, the FDBK (feedback) field may indicate error or completion status about the write request. (See Appendix A for a description of the feedback field.)

Register 15: One of the following hexadecimal values is indicated:

- 0 If the ASY option code is in effect, VTAM found no errors or contradictions in the way that the write request was made and has therefore accepted the request. If the SYN option code is in effect, the write operation has been completed successfully.
- 4 The request cannot be accepted because the RPL is being used by another request, or the terminal is not connected to your application program. The RPL's FDBK field has not been set. If an active LERAD exit routine exists, it has been invoked.
- 8 A logical error has occurred; the FDBK field can be examined to determine which one it was. If an active LERAD exit list routine exists, it has been invoked. (This return code can be posted only if the SYN option code is in effect.)
- C A physical error has occurred; the FDBK field can be examined to determine which one it was. If an active SYNAD exit routine is available, it has been invoked. (This return code can be posted only if the SYN option code is in effect.)
- 10 A conversational write operation was requested, but the terminal did not respond with data. Since the LBM option code is in effect, the conversational write operation is considered completed.
- 14 A special condition involving the terminal occurred. These include error lock set, RVI line control character received, and attention interruption detected. The FDBK field indicates the specific condition.
- 18 The operation was canceled by a RESET request.
- 1C VTAM canceled the operation; the second byte of the FDBK field is set indicating the reason.

Appendix A: Interpreting the Feedback Field

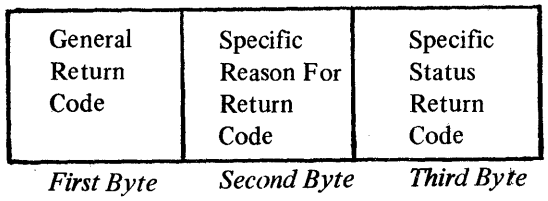
FDBK, a three-byte field in the RPL control block, is used by VTAM to describe how a requested operation was completed. FDBK posting applies to those operations requested by any of the VTAM macro instructions that directly refer to an RPL (that is, that have RPL=rpl address as an operand).

Two steps are involved in testing a FDBK field's contents. The first step is to extract the fields contents from the RPL. This can be done with the SHOWCB macro instruction. For example:

```
SHOWCB    RPL=RPL1,                                    X
          FIELDS=FDBK,                                 X
          WAREA=FDBKAREA,                              X
          LENGTH=4
```

FDBK will be placed in the right-most three bytes of FDBKAREA. After FDBK has been removed from the RPL, the next step is to analyze the FDBK information. This can be accomplished with assembler instructions using whatever technique the programmer prefers -- testing under mask, register shifting, and so forth.

The following box summarizes the kind of information that is indicated in each byte.



The general return code indicates whether the operation was completed successfully or, if it was not, what category of error occurred.

0 The requested operation was completed successfully. The contents of the third byte (the specific status return code) is therefore valid and should be checked.

4 Reserved.

8 A *logical* error occurred. A logical error results from an inherent contradiction in a request -- like writing to a terminal not currently connected to your application program, or refering in an I/O request to an ACB that has never been opened.

Logical errors, unlike physical errors, result from flaws in the way a program is written. There is, therefore, no point in using the LERAD exit list routine (which is scheduled when a logical error occurs) to attempt to correct the problem during program execution. The LERAD exit list routine should instead be used to obtain a program dump, so that the logical error can be identified and the program corrected.

C A *physical* error occurred. A physical (or "I/O") error is an error that results from a hardware malfunction or a line disconnection that prevents successful completion of the requested operation. The programmer can prepare routines to handle the occurrence of physical errors. These routines would be part of the SYNAD exit list routine, which is scheduled when a physical error occurs.

10 A conditional request has been completed with no action taken, since the condition was not met. A conditional RESET request, for example, cancels an I/O operation only if the operation is pending; if data transfer has begun, no cancellation can be performed and this return code is set. This return code is also set if the NQ option code is in effect for an OPNDST request, and the indicated terminal is not available.

14 A special condition involving the terminal occurred. These include error lock set, RVI line control character received, and attention interruption detected. The second byte indicates the specific condition.

18 A RESET request canceled the operation.

1C The operation was canceled because of an action external to your program. These actions include an error in the Network Control Program (NCP), a permanent channel failure, and a quick closedown initiated by the network operator. The second byte indicates the specific action that occurred.

Second Byte
The information conveyed by the second byte depends on the general return code contained in the first byte.

When the general return code is 8 (logical error), the specific return code indicates problems such as these:

Invalid request format.

Data length incompatible with receiving device.

The ACB has not been opened.

The terminal currently being solicited is not connected to your application program.

The terminal being written to is an output-only terminal.

INTRPRET could not find the requested information.

When the general return code is C (physical error), one *or more* of the following problems are indicated:

A physical error occurred *and* an error lock has been set for the device.

A physical error occurred *and* an RVI line control character has been received.

A physical error occurred *and* an attention interruption has been detected (2741 and 1050 terminals only).

A physical error occurred; the operation was retried without success, and the error has been classified as permanent (device not useable).

A physical error occurred during an output operation.

A physical error occurred that involves a channel or control unit.

When the general return code is 14 (special condition), one *or more* of the following problems are indicated:

An error lock has been set for the device.

An RVI line control character has been received.

An attention interruption has been detected.

When the general return code is 1C (VTAM-canceled operation), one of the following causes are indicated:

A temporary failure occurred in the Network Control Program (NCP).

A failure occurred in the Network Control Program from which recovery was unsuccessful. No further communication with the device is possible.

A permanent channel failure occurred.

The network operator initiated a quick closedown.

Third Byte

The specific status return code should be checked upon completion of I/O requests when the general return code indicates successful completion (0 in the first byte). The following list shows some typical status conditions. Note that the status conditions indicated here are not exclusive of each other; several may be indicated:

The block of data just received was not the last block of a message or transmission.

The block of data just received was the last block of a message.

Either the block of data just received was the last block of a transmission, or an EOT unaccompanied by text was received.

A conversational reply is now possible (that is, the block of text just received will not be acknowledged until a new request is directed at the sending device).

Leading graphic characters have been received.

A block of heading characters has been received.

Appendix B: Line Control Characters Recognized or Sent by VTAM Macro Instructions

The first three columns in the box show the line- control character that delimits 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 line control characters added to the beginning and end of the user-supplied data when a WRITE or DO 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			Writing		
	BLOCK	MSG	TRANS	(b) = inserted at the beginning (e) = inserted at end		
				BLK	LBM	LBT
IBM 1050 Data Communication System	EOB	EOT [1]	EOT [1]	EOA(b) EOB(e)	EOA(b) EOB(e) [2]	EOA(b) EOB(e) [2]
IBM 2740 Communication Terminal, Model 1	EOT [1]	EOT [1]	EOT [1]	EOA(b) NULL(e)	EOA(b) NULL(e)	EOA(b) NULL(e)
IBM 2740 Communication Terminal, Model 1, with checking	EOB	EOT [1]	EOT [1]	EOA(b) EOB(e)	EOA(b) EOB(e) [2]	EOA(b) EOB(e) [2]
IBM 2740 Communication Terminal, Model 1, with checking and station control	EOB	EOT [1]	EOT [1]	EOA(b) EOB(e)	EOA(b) EOT(e)	EOA(b) EOT(e)
IBM 2740 Communication Terminal, Model 2	EOT [1]	EOT [1]	EOT [1]	EOA(b) EOT(e)	EOA(b) EOT(e)	EOA(b) EOT(e)
IBM 2741 Communication Terminal	EOT [1]	EOT [1]	EOT [1]	EOA(b) NULL(e)	EOA(b) NULL(e)	EOA(b) NULL(e)
IBM Communication Magnetic Card Selectric Typewriter	EOT [1]	EOT [1]	EOT [1]	EOA(b) NULL(e)	EOA(b) NULL(e)	EOA(b) NULL(e)
IBM World Trade Telegraph Station	EOT [1]	EOT [1]	EOT [1]	CCITT header	CCITT header	CCITT header
IBM SYSTEM/7	EOT [1]	EOT [1]	EOT [1]	EOA(b) NULL(e)	EOA(b) NULL(e)	EOA(b) NULL(e)
AT&T 83B3 Selective Calling Station	EOT [1]	EOT [1]	EOT [1]	none(b) EOM(e)	none(b) EOM(e)	none(b) EOM(e)
AT&T Teletypewriter Terminal, Models 33 and 35	EOT [1]	EOT [1]	EOT [1]	none(b) EOM(e)	none(b) EOM(e)	none(b) EOM(e)
Western Union Plan 115A Station	EOT [1]	EOT [1]	EOT [1]	none(b) EOM(e)	none(b) EOM(e)	none(b) EOM(e)

[1] This EOT will be either a circle C or an equivalent teletype sequence.

[2] And an EOT will be sent when the block is acknowledged by the system with a positive response.

Binary Synchronous Devices:	Soliciting			Writing		
	BLOCK	MSG	TRANS	(b) = inserted at the beginning (e) = inserted at end		
				BLK	LBM	LBT
IBM 2770 Data Communication Terminal	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) [3]
IBM 2780 Data Transmission Terminal	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) [3]
IBM 2972 General Banking Terminal, Models 8 and 11	ETB	ETX	EOT	STX(b) ETX(e)	STX(b) ETX(e)	STX(b) ETX(e) [3]
IBM 3270 Information Display System, locally attached.			EOT	[4]	[4]	[4]
IBM 3270 Information Display System, remotely attached			EOT	[4]	[4]	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) [3]
IBM 3740 Data Entry System	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) [3]
IBM 3780 Data Transmission Terminal	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) [3]
IBM SYSTEM/3	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) [3]
IBM SYSTEM/370	ETB	ETX	EOT	STX(b) ETB(e)	STX(b) ETX(e)	STX(b) ETX(e) [3]

[3] And an EOT will be sent when the block is acknowledged by the system with a positive response.

[4] No line control characters will be sent. Write control characters must be in the data.

Appendix C: Summary of Control Block Field Usage

Once you are familiar with the workings of the VTAM macro instructions described in this book, this appendix can be used as a quick-reference source. The appendix 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 for a given use of a macro instruction, not all fields necessarily *will* apply. Refer back to the macro instruction descriptions if you are in doubt.

The following paragraph explains what the information supplied with the first macro instruction means. Compare this explanation with the first macro instruction shown below (CHANGE); by way of example, this will show you how to interpret the information supplied with all of the macro instructions.

When the application program issues a CHANGE macro instruction, the ACB field of CHANGE's RPL must contain the address of an ACB ("ACB field → ACB") and the NIB field must contain the address of the modified NIB ("NIB field → modified NIB"). The appropriate option codes must be set in the OPTCD field ("OPTCD field = option codes"). The ECB or EXIT fields, if used, must contain the address of a fullword area or an exit routine, respectively ("ECB field → fullword work area" and "EXIT field → exit routine"). Note that information supplied above the dashed line shows fields set by the *application program*, and information shown below the dashed line shows fields and registers set by *VTAM*. Also note the use of the pointer (→) and the equals sign. "ABC field → ACB" means that the ACB field must *point to* (that is, contain the address of) an ACB. "FDBK field = status information" on the other hand, means that the FDBK field *contains* the status information.

CHANGE → RPL: ACB field → ACB
NIB field → modified NIB
{ECB field → fullword work area
{EXIT field → RPL exit routine
OPTCD field = option code

Register 15 = return code
RPL: FDBK field = status information

CHECK → RPL being checked

Register 15 = return code

CLOSE → ACB being closed

Register 15 = return code
ACB: ERROR field = specific error status information

CLSDST → RPL: ACB field → ACB
{NIB field → NIB: NAME field = symbolic name of terminal to be disconnected
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
EXIT field → RPL exit routine}
OPTCD field = option codes

Register 15 = return code
RPL: FDBK field = status information

DO → RPL: ACB field → ACB
ARG field = CID of terminal
{ECB field → fullword work area
EXIT field → RPL exit routine}
OPTCD field = option codes
AREA field → LDO: If CMD field = COPYLBM or COPYLBT,
ADDR → copy control char. 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 = WRITE, WRITELBM, WRITELBT, WRTHDR, WRTNRLG, or WRTPRLG,
ADDR → output data
LEN = length of output data

Register 15 = return code
RPL: AAREA field → last LDO used
USER field → data from USERFLD field of NIB
RECLEN field = length of data received
FDBK field = status information

GENCB 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 = standard, list, or execute form designation

Register 0 = length of control blocks
Register 1 → generated control blocks, if built in dynamically allocated storage
Register 15 = return code

INQUIRE → RPL: ACB field → ACB
{ECB field → fullword work area
EXIT field → RPL exit routine}
OPTCD = option codes

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, as defined by the GROUP 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 = 8
If OPTCD = APPSTAT,
NIB field → NIB: NAME field = symbolic name of application program
AREA field → input area for data
AREALEN field = 4
If OPTCD = CIDXLATE,
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

Register 15 = return code
RPL: RECLEN field = length of data received
FDBK field = status information

INTRPRET → RPL: ACB field → ACB
{NIB field → NIB: NAME field = symbolic name of terminal
ARG field = CID of terminal}
AREA field → logon message
RECLEN field = length of logon message
AAREA field → input data area
AAREALN field = length of input data area
{ECB field → fullword work area
EXIT field → RPL exit routine}
OPTCD field = option codes

Register 15 = return code
RPL: ARECLEN field = length of data received
FDBK field = status information

MODCB control block type operand → control block
control block field name operand = new value to be set
MF operand = standard, list, or execute form designation
AM = VTAM

Register 15 = 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 → ACB
{ECB field → fullword work area
EXIT field → RPL exit routine}
OPTCD field = option codes
NIB field → NIB: PROC field = processing options
If OPTCD = ACQUIRE,
NAME field = symbolic name of terminal
If OPTCD = ACCEPT and ANY,
NAME field not examined
If OPTCD = ACCEPT and SPEC,
NAME field = symbolic name of terminal

Register 15 = return code
RPL: ARG field = CID of connected terminal
AREA field → NIB: CID field = CID of terminal
USER field = data from NIB's USERFLD field
FDBK field = status information

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
EXIT field → RPL exit routine}
OPTCD field = option codes

Register 15 = return code
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

RESET → RPL: ACB field → ACB
ARG field = CID of terminal
{ECB field → fullword work area
EXIT field → RPL exit routine}
OPTCD field = option codes

Register 15 = return code
RPL: USER field = data from USERFLD field in NIB
FDBK field = status information

SETLOGON → RPL: ACB field → ACB
{ECB field → fullword work area
EXIT field → RPL exit routine}
OPTCD field = option codes

Register 15 = return code

SHOWCB

control block type operand → control block
FIELDS operand = fields to be moved
AREA operand → work area where fields will be moved
LENGTH operand = length of work area
MF operand = standard, list, or execute form designation
AM = VTAM

Register 0 = required length for work area (if work area too small)
Register 15 = return code

SIMLOGON → RPL: ACB field → ACB
NIB field → NIB: NAME field = symbolic name of terminal
PROC field = processing options
AREA field → logon message
RECLEN field = length of logon message
{ECB field → fullword work area
EXIT field → RPL exit routine}
OPTCD field = option codes

Register 15 = return code
RPL: FDBK field = status information

SOLICIT → RPL: ACB field → ACB
ARG field → CID of source terminal (if OPTCD = SPEC)
{ECB field → fullword work area
EXIT field → RPL exit routine}
OPTCD field = option codes

Register 15 = return code
RPL: USER field = data from USERFLD field of NIB
FDBK field = status information

TESTCB

control block type operand → control block
field name operand = test value (or IO = COMPLETE)
ERET operand → error exit routine
MF operand = standard, list, or execute form designation
AM = VTAM

Register 15 = 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
EXIT field → RPL exit routine}
OPTCD field = option codes

Register 15 = return code
RPL: ARECLEN field = length of input data
USER field = data from USERFLD field of NIB
FDBK field = status information

Glossary

ACB: *Access method control block.*

acceptance: The process by which a teleprocessing program connects a node in response to a logon request from that node. It is implemented by an OPNDST macro instruction having the ACCEPT option code set in its RPL.

access method control block: In VTAM, a control block that links a teleprocessing program to VTAM. Abbreviated *ACB.*

acquisition: The process by which a teleprocessing program initiates and secures connection to another node. It is implemented by an OPNDST macro instruction having the ACQUIRE option code set in its RPL.

active exit list routine: In VTAM, a routine whose address has been placed in an exit list (EXLST) control block and marked active.

active RPL: An RPL that is in use. For synchronous requests, and RPL is active until the request is completed. For asynchronous requests, an RPL is active until it is checked with a CHECK macro instruction.

advanced telecommunication access method: A set of IBM programs that control communication between terminals and teleprocessing programs running under operating systems with virtual storage.

application program: (1) To *VTAM,* the requests and control blocks that refer to a given ACB, or are pointed to by that ACB. To VTAM and the rest of the telecommunications network, the ACB represents the application program. (2) To the *programmer,* the program code that performs a given function. This code includes everything in definition (1) above, plus: all of the macro instructions that build and manipulate the control blocks in definition (1), and all of the non-VTAM instructions that evaluate the input data received by the program and prepare the output data to be sent from it. (3) To the *operating system,* the program code that has been organized into a job step. This code includes everything in definition (2) above, but could also include several multiples of everything in definition (2).

For example: A given ACB may be linked to an entry in the resource definition table called PAYROLL1. To VTAM, all requests and control blocks associated with the PAYROLL1 ACB constitute an application program. The programmer will consider these requests and control blocks, *and* the program code needed to support these requests and to build and manipulate these control blocks, as the application program PAYROLL1. The programmer could use common supporting code for several ACBs – PAYROLL1, PAYROLL2, and PAYROLL3, for example. The programmer might think of these three as one application program or perhaps as three. VTAM would consider them as three separate application programs. If these three were combined into one job step, they would clearly be one application program to the operating system.

application program identification: In VTAM, the symbolic name by which a teleprocessing program is identified to VTAM and the rest of the teleprocessing network. This name appears in the application program's ACB and in the resource definition table.

asynchronous exit list routine: An exit list routine that is scheduled as a result of events that are not initiated by the application program, but are instead imposed from outside of the application program's control. For example, the RELREQ exit list routine is scheduled when an application program requests connection to a terminal already connected to another application program. All exit list routines are asynchronous exit list routines except the LERAD and the SYNAD exit list routines.

asynchronous request: A request that causes control to be returned to the application program *before* the requested operation is completed. When the operation is completed, VTAM either invokes the RPL exit routine, or posts an ECB (in which case the program must issue a CHECK macro instruction to determine whether posting has occurred). A request is made asynchronous be setting the ASY option code in its RPL. Contrast with synchronous request.

automatic logon request: A logon request, to a specified application program, generated by VTAM (rather than by the terminal itself) when the terminal becomes available for connection. Automatic logon requests may be specified by the installation during VTAM definition.

block: (1) A group of bits, or n-ary digits, transmitted as a unit. (2) The smallest complete unit of data that may be transmitted between a teleprocessing program and a connected terminal. The maximum size of a block is determined by the characteristics of the device that is sending or receiving the data.

CID: VTAM's shortened form of a terminal's symbolic name. The installation assigns a symbolic name to each terminal (or dial-up line) in its network configuration. When the application program requests connection to the terminal – by placing the terminal's symbolic name into a NIB and issuing an OPNDST macro instruction – VTAM converts this eight-byte symbolic name into a four-byte CID (communications ID). The CID is placed in the NIB and in OPNDST's RPL. The application program must use this CID for all subsequent communication requests for the terminal.

closedown: In a telecommunication system, the orderly deactivation of a telecommunication access method and network. In VTAM, a normal closedown does not take effect until all application programs have disconnected their terminals and closed their ACBs. Until then, all data-transfer operations continue, but VTAM rejects any further logon requests.

communications controller: A type of communication control unit whose operations are controlled by a program stored and executed in the unit.

connect: In VTAM, to establish and prepare a network path for communication between two nodes.

conversational write operation: A composite operation wherein data is first sent to a terminal, and then data is read from that terminal. It is implemented with a WRITE macro instruction having the CONV option code set in its RPL.

data transfer: Same as *data transmission.*

data transmission: In telecommunications, the sending of data from one node to another.

device-control character: A control character that is embedded in a data stream to control mechanical and format operations at a terminal (for example, a line-feed character or carriage-return character). Contrast with *line-control character.*

dialog: In VTAM, a series of data exchanges between an application program and a terminal, during which the terminal has scheduling priority over other terminals on the same multipoint line. (Scheduling priority is explained under the ED-CD option code in the RPL macro instruction description.) A dialog is approximately equivalent to a session in the 3704 or 3705 Network Control Program.

disconnect: In VTAM, to suspend use of a network path between two connected nodes.

error lock: A condition established by the operating system or by a 3704 or 3705 communications controller wherein communication with the terminal is suspended. Error locks can be set for a terminal when the communications controller detects conditions such as these:

Unrecoverable hardware malfunctions

Negative polling limits exceeded

Attention interruptions received

The error lock is set when the system or the controller detects an unrecoverable hardware malfunction, or if the device's operator causes an attention interruption.

exit list: In VTAM, a control block that contains the names of teleprocessing-program routines that receive control when specified events occur during VTAM execution. For example, programs named in the exit list handle such conditions as logon processing and I/O errors. Abbreviated *EXLST.*

exit list routine: A routine whose address has been placed in an exit list (EXLST) control block. The addresses are placed there with the EXLST macro instruction, and the routines are named according to their corresponding operand; hence SYNAD exit list routine, LERAD exit list routine, UNSIP exit list routine, and so forth. All exit list routines are coded by the application programmer. Contrast with RPL exit routine.

EXLST: *Exit list*

inactive node: In VTAM, a node that is not attached to VTAM and is not available for connection to another node.

input operation: In VTAM, a read or solicit operation. Input requests are implemented by READ and SOLICIT macro instructions, and by the conversational variation of WRITE macro instructions.

leading graphics: From one-to-seven graphic characters that may accompany an acknowledgment sent to a binary synchronous terminal in response to receipt of a block of data.

line: The communication medium linking a communication control unit to another communication control unit, or linking a communication control unit to one or more terminals.

line-control character: A character in a data stream that controls the transmission of data over a network path; for example, line-control characters delimit messages and indicate whether a node has data to send or is ready to receive data.

line-control discipline: A general term for the set of rules, requirements, and procedures for transmitting information to and from a particular type of terminal in a telecommunication system.

line group: A set of one or more lines of the same type by which terminals are attached to a communication control unit.

local: Pertaining to terminals and communication control units that are attached directly by channels to a central computer.

logical device order: In VTAM, a set of parameters that specify a data-transfer or data-control operation.

logical error: An error that results from a VTAM request that is self-contradictory.

logoff request: A request by a terminal user to be disconnected from a teleprocessing program.

logon request: A request initiated by a device, for connection between itself and an application program. (Contrast with automatic logon requests and simulated logon requests, which are initiated by a program on behalf of a device.)

message: In telecommunications, a combination of characters representing one or more blocks that form a logical entity.

negative polling limit: In the 3704 or 3705 Network Control Program, the maximum number of consecutive negative responses to polling that the Network Control Program will accept before breaking off communication.

NIB list: In VTAM, a list of contiguous NIBs (node initialization blocks) that describe nodes that are to be connected as a group.

node: In VTAM, an addressable point in a telecommunication system. Nodes include terminal components, terminal control units, teleprocessing programs, and remote computers.

node initialization block: A control block that is associated with a particular node, and contains information used by the application program to identify a node and indicate how communication requests directed at the node are to be implemented. Abbreviated NIB.

node name: In VTAM, the symbolic name associated with a specific node and assigned during network definition.

option code: One of the indicators set by the OPTCD operand of the RPL macro instruction. These indicate how a given communication request is to be implemented by VTAM.

physical error: An error that results from unforeseen hardware errors occurring in the teleprocessing network, such as terminal malfunctions or line transmission failures.

processing option: One of the indicators set by the PROC operand of the NIB macro instruction. These indicate how communication requests are to be implemented for a given terminal.

quick closedown: In VTAM, a closedown in which current data-transfer operations are completed, while pending data-transfer requests are canceled.

RDT: *Resource definition table*

read operation: The transference of data from VTAM buffers to program storage.

read request: Any request for a read operation. There are two such requests:

A READ macro instruction.

A WRITE macro instruction, if the CONV option code is in effect for the request's RPL.

remote: Pertaining to terminals and communication control units that are attached to a central computer through a communication control unit.

request parameter list: A control block that contains the parameters necessary for processing a request for data transfer or a request for connecting or disconnecting a node. Abbreviated RPL.

resource definition table: In VTAM, a table that describes the characteristics of each node available to VTAM, and associates each node with an address. The resource definition table is built during VTAM definition with APPL, LINE, GROUP, and TERMINAL macro instructions, but it can be modified by the network operator while VTAM is running.

RPL: *Request parameter list*

RPL exit routine: A routine whose address has been placed in the EXIT field of an RPL. For asynchronous requests, this routine is automatically invoked by VTAM when the request associated with the RPL is completed. Contrast with exit list routine.

session: (1) In VTAM, the period of time during which a terminal is connected to an application program. (2) In a 3704 or 3705 Network Control Program, a series of command and data interchanges between the host processor and a teleprocessing device.

session limit: In a 3704 or 3705 Network Control Program, the maximum number of concurrent sessions that can be initiated on a multipoint line (or point-to-point line where the terminal has multiple components.)

shared: (1) Pertaining to communication control units, network paths, and communication lines that may be used concurrently by several teleprocessing programs to communicate with different nodes. (2) Pertaining to terminals that may be used by more than one teleprocessing program; only one teleprocessing program may be connected to a shared terminal at any one time.

simulated logon request: A request initiated by a program (via the SIMLOGON macro instruction) on behalf of a device, for connection between the device and a program. Contrast with *logon request* and *automatic logon request.*

solicit operation: The process of obtaining (or attempting to obtain) data from a device and moving that data into VTAM buffers.

solicit request: Any request for a solicit operation. There are two such requests:

A SOLICIT macro instruction.

A read request, if the SPEC option code is in effect, and if VTAM buffers currently hold no data from the device being read from.

synchronous exit list routine: An exit list routine that is scheduled as a result of an operation requested by the application program. There are only two such routines – the LERAD and the SYNAD exit list routines – all of the other exit list routines are scheduled as the result of events not connected with requests initiated by the application program. The latter are *asynchrouous* exit list routines.

synchronous request: A request that will cause control to be returned to the application program only *after* the requested operation has completed. A request is made synchronous by setting the SYN option code in its RPL. Contrast with asynchronous request.

telecommunication network: The complex of all terminals and communication devices, and the lines and channels that connect them to one another and to a central processing system.

teleprocessing program: In VTAM, a program, treated as a node, that uses the services of VTAM to communicate with local and remote terminals.

teleprocessing system: A general term for a complex of interconnected central computers, communication devices, and programs.

terminal: A node in a telecommunication network at which data can enter or leave. A terminal can be an input/output device or a terminal control unit to which one or more input/output devices (terminal components) are attached. Terminals also include remote computers, when they are performing as I/O devices.

terminal component: A separately addressable part of a terminal that performs input or output functions.

transmission: A logical group of one or more messages.

transmission limit: In the 3704 or 3705 Network Control Program, the maximum number of transmissions that can be sent to or received from a teleprocessing device during one session on a multipoint line (or point-to-point line where the terminal has multiple components) before the Network Control Program suspends the session to service other devices on the line.

transparent mode: A mode of binary synchronous transmission in which all data, including normally restricted line-control characters, is transmitted only as bit patterns. Control characters that are intended to be effective are preceded by a DLE character.

VTAM definition: The process of (1) including VTAM in the operating system generation (SYSGEN), (2) defining the teleprocessing network to VTAM and 3704 or 3705 Network Control Program, and (3) modifying IBM- defined VTAM characteristics to suit the needs of the installation. VTAM definition is implemented by the installation with definition macro instructions and operator commands.

write operation: The transference of data from program storage to a device.

write request: Any request for a write operation. Write requests are implemented with the WRITE macro instruction.

Index

GC27-6995-0

VTAM APRM (File No. S370-30) Printed in U.S.A. GC27-6995-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)