SPERRY UNIVAC

VTAM
Programmer Reference

Mini-Computer Operations

2722 Michelson Drive
P.O. Box C-19504
Irvine, California 92713
98A 9952 224

VTAM
PROGRAMMER REFERENCE MANUAL

98A 9952 224
FEBRUARY 1978

The statements in this publication are not intended to create any warranty, express or implied. Equipment specifications and performance characteristics stated herein may be changed at any time without notice. Address comments regarding this document to Sperry Univac, Mini-Computer Operations, Publications Department, 2722 Michelson Drive, P.O. Box C-19504, Irvine, California, 92713.

© 1978 SPERRY RAND CORPORATION

Sperry Univac is a division of Sperry Rand Corporation

Printed in U.S.A.

CHANGE RECORD

Page Number	Issue Date	Change Description
all	2/77	Minor revisions have been incorporated throughout this manual.
various	2/78	Minor revisions incorporated throughout the manual, and references to Varian deleted.

Change Procedure:

When changes occur to this manual, updated pages are issued to replace the obsolete pages. On each updated page, a vertical line is drawn in the margin to flag each change and a letter is added to the page number. When the manual is revised and completely reprinted, the vertical line and page-number letter are removed.

96A0730-000A

LIST OF EFFECTIVE PAGES

Page Number	Change in Effect	Page Number	Change in Effect
all	completely revised		

TABLE OF CONTENTS

SECTION 1
INTRODUCTION

1.1 INTRODUCTION 1-1
1.2 SYSTEM FLOW AND ORGANIZATION 1-1
1.3 HARDWARE SUPPORTED AND REQUIRED 1-3
1.4 GUIDE TO THIS MANUAL 1-3
1.5 BIBLIOGRAPHY 1-4

SECTION 2
DEFINING A COMMUNICATIONS NETWORK

2.1 INTRODUCTION 2-1
2.1.1 Input to the NDM 2-1
2.1.2 General Format 2-1
2.2 NETWORK DEFINITION LANGUAGE STATEMENTS 2-1
2.2.1 LINE Statement 2-1
2.2.2 TERMINAL Statement 2-4
2.2.3 END Statement 2-5
2.3 OPERATING INSTRUCTIONS 2-6
2.4 ERROR INDICATIONS AND WARNINGS 2-6
2.5 NDM OUTPUT 2-6

SECTION 3
USING VTAM MACROS

3.1 INTRODUCTION 3-1
3.2 GENERAL FORM 3-1
3.3 ERROR INDICATIONS ON VTAM MACROS 3-2

SECTION 4
OPENING AND CLOSING TERMINALS AND LINES

4.1 INTRODUCTION 4-1
4.2 OPEN MACRO AND JCP DIRECTIVE 4-1
4.2.1 Forms of OPEN Macro 4-1
4.2.2 Error Indications on OPEN 4-2
4.3 CLOSE MACRO AND JCP DIRECTIVE 4-3
4.3.1 General Format 4-3
4.3.2 Error Indications 4-3

SECTION 5
PROGRAMMING AT TCM LEVEL

5.1 MACRO DEFINITION 5-1
5.2.1 READ Macro 5-1
5.2.2 WRITE Macro 5-2
5.2.3 STAT Macro 5-3
5.2.4 FUNC Macro 5-4
5.2.5 WEOF Macro 5-6
5.3 TTY TCM WITH DIAL-UP LINES 5-6
5.4 FORTRAN LEVEL PROGRAMMING 5-6

SECTION 6
PROGRAMMING AT THE CCM LEVEL

6.1 INTRODUCTION 6-1
6.2 CCM I/O CONTROL MACROS AND FUNCTIONS 6-1
6.2.1 LCB Macro 6-1
6.2.2 OPEN Macro 6-3
6.2.3 CLOSE Macro 6-3
6.2.4 READ Macro 6-3
6.2.5 WRITE Macro 6-3
6.2.6 FUNC Macro 6-4
6.2.7 STAT Macro 6-7

SECTION 7
BUFFER CHAINING

7.1 INTRODUCTION 7-1
7.1.1 Queuing Procedure 7-1
7.1.2 PUTQ 7-1
7.1.3 GETQ 7-1
7.2 CHAIN HEADER 7-2
7.3 INTERFACE BLOCK HEADER 7-2
7.4 SET AND RESET FUNCTIONS 7-4
7.5 PROCEDURE FOR CODING A BUFFER CHAIN 7-5

SECTION 8
BINARY SYNCHRONOUS COMMUNICATION

8.1 INTRODUCTION 8-1
8.2 DATA LINK 8-1
8.2.1 Point-To-Point Data Link 8-1

SECTION 8
BINARY SYNCHRONOUS COMMUNICATION *(continued)*

8.2.2 Multipoint Data Link 8-1
8.3 TRANSMISSION CODES 8-1
8.4 OPERATION OF THE DATA LINK 8-2
8.4.1 Polling and Selection 8-2
8.4.2 Message Blocks 8-3
8.4.3 Error Checking 8-3
8.4.4 EOT/NAK Pad Format Check 8-4
8.4.5 Data Link Control 8-4
8.5 MESSAGE FORMATS 8-6
8.5.1 Initialization Procedure 8-6
8.5.2 Message Transfer Procedure 8-8
8.5.3 Termination Procedure 8-8
8.5.4 Transparent Mode 8-8
8.5.5 Timeouts 8-9
8.5.6 Pad Characters 8-10
8.6 TRANSMISSION SEQUENCE AND RECOVERY PROCEDURES 8-10

SECTION 9
MANAGING BUFFERS

9.1 INTRODUCTION 9-1
9.2 MEMORY ALLOCATION ROUTINES AND THEIR FUNCTIONS 9-1
9.2.1 VT$BMT 9-1
9.2.2 VT$GTM 9-2
9.2.3 VT$PTM 9-2

SECTION 10
CODING A TERMINAL CONTROLLER MODULE (TCM) FOR VTAM

10.1 INTRODUCTION 10-1
10.2 TABLES USED BY TCM 10-1
10.3 TCM FUNCTIONS 10-2
10.4 TCM COMPONENTS 10-3
10.5 MODIFYING THE NETWORK DEFINITION MODULE 10-5
10.6 PROCEDURE TO CODE A TCM FOR VTAM 10-6

SECTION 11
CONTROLLING A NETWORK

11.1 INTRODUCTION 11-1
11.2 DIRECTIVES 11-1
11.2.1 General Format of NCM Directives 11-1
11.2.2 UP Directive 11-1
11.2.3 DOWN Directive 11-1
11.2.4 REDIRECT Directive 11-2
11.2.5 RESTORE Directive 11-2
11.2.6 LIST Directive 11-3

SECTION 12
PROGRAMMING AN APPLICATION

SECTION 13
CONFIGURING A VTAM SYSTEM

13.1 INTRODUCTION 13-1
13.2 MODIFYING VTAM CCM TABLES AND ADDING CONTROLLER TABLES 13-1
13.2.1 CCM Tables 13-1
13.2.2 Controller Table 13-2
13.3 ADDING TDF RECORDS FOR VTAM CCM's 13-2
13.4 ADDING TDF RECORDS FOR TCM (TTY) 13-2
13.5 RESERVING MEMORY 13-2
13.6 DEFINING PERIPHERAL ARCHITECTURE 13-3
13.7 DEFINING INTERRUPT STRUCTURE 13-3
13.8 ASSIGN LOGICAL UNITS TO PHYSICAL DEVICES 13-4
13.9 LOADING ANCILLARY VTAM MODULES 13-4
13.10 VTAM MEMORY REQUIREMENTS' 13-4

APPENDIX A
TELETYPE AND CRT CHARACTER CODES

APPENDIX B
EBCDIC AND ASCII CHARACTER ASSIGNMENTS

INDEX

LIST OF ILLUSTRATIONS

Figure 1-1. Structure of VTAM 1-2
Figure 1-2. Data Flow from Application to Terminal 1-3
Figure 1-3. Input and Output to Network Definition Module 1-3
Figure 7-1. Contents of CHR and IBHs after PUTQ 7-3
Figure 7-2. Contents of CHR and IBHs During READ 7-4
Figure 7-3. Contents of CHR and IBHs Before and After GETQ 7-4
Figure 7-4. Relationship of CHR and IBHs 7-5
Figure 8-1. Regular Message Format 8-3
Figure 8-2. Error Checking Capabilities 8-4
Figure 8-3. Use of WACK, RVI, and TTD 8-7
Figure 8-4. Transparent Data Block 8-9
Figure 10-1. VTAM TCM and TTY TCM Modules 10-4
Figure 12-1 Flowchart of VTAM Application 12-1

LIST OF TABLES

Table 2-1. LSD Field Description and Range 2-7
Table 2-2. TIB Field Description and Range 2-8
Table 2-3. TCD Field Description and Range 2-8
Table 3-1. Detail Status 3-2
Table 8-1. Control Characters 8-4
Table 8-2. Transmission and Recovery Procedures 8-11
Table 13-1. Direct Connect Interrupts 13-3

SECTION 1
INTRODUCTION

The Vortex Telecommunications Access Method (*VTAM*) provides teleprocessing controls for communications controllers, modems, terminals, communications networks and network-operator interfacing. VTAM is an integral part of the VORTEX operating system. It extends the capabilities of the real-time multi-tasking operating system into the growing area of telecommunications.

Through the combination of VTAM and VORTEX access to remote devices is as simple as that for on-site computer peripherals. VTAM gives the user the same format for requests for telecommunications as is available for printers and magnetic-tape units.

At the same time, the user is assured of an open-ended system design that can accomodate his future requirements. VTAM is modular in its structure and so provides a software foundation on which to build systems tailored to their applications.

In summary VTAM provides

- a standard subsystem under VORTEX without affecting the utility of VORTEX in other applications
- phased implementation to allow changes for new equipment and expansion
- modularity in structure to satisfy diverse requirements
- interfaces for applications to be removed from handling line and terminal characteristics
- a simplified method of configuring lines and terminals through the Network Definition Language
- VTAM tasks a user can call to allocate memory dynamically
- an optional, automatic buffer chaining on input
- on-line query and control of communication system status

1.2 SYSTEM FLOW AND ORGANIZATION

The three modules which are the basic building blocks of a VTAM System are the communications controller, terminal control, and network control modules. The most basic VTAM component, the Communications Controller Module (CCM), drives a multiplexor or controller hardware. The Terminal Control Module (TCM) provides an optional level of control for terminals and lines. TCM's handle such items as terminal errors and line adaptor control. The Network Control Module (NCM) furnishes an interface with the network for the computer operator.

Figure 1-1 is an overview of the flow in a VTAM system.

The flow of data to an application program under VTAM and VORTEX is first under control of a CCM. The incoming data from the line is initially handled by the LAD and the multiplexor and packed into a buffer. If the READ request is directed to the terminal, a TCM then converts, formats and segments the data. If required for the terminal type, the TCM could provide terminal control procedures. The user can bypass this level and provide his own terminal-oriented procedures in the application and pass his input and output request directly to the CCM (see figure 1-2).

Features of VTAM Modules

TCM stands for Terminal Control Module

- Interfaces with application through standard request
- Establishes terminal disciplines and line protocol
- Converts codes and formats data for terminals
- Compresses and decompresses data
- Performs modem control functions
- Operates independent of type of controller

CCM stands for Communications Controller Module

- Provides a common interface for all TCM's
- Performs mechanics of data input and transmission
- Handles all controllers of one type
- Operates transparently with respect to terminal type

NCM stands for Network Control Module

- Provides dynamic network control
- Allows alternate line or terminal selection
- Provides inquiry about status of lines and terminals
- Allows setting lines and terminals UP or DOWN

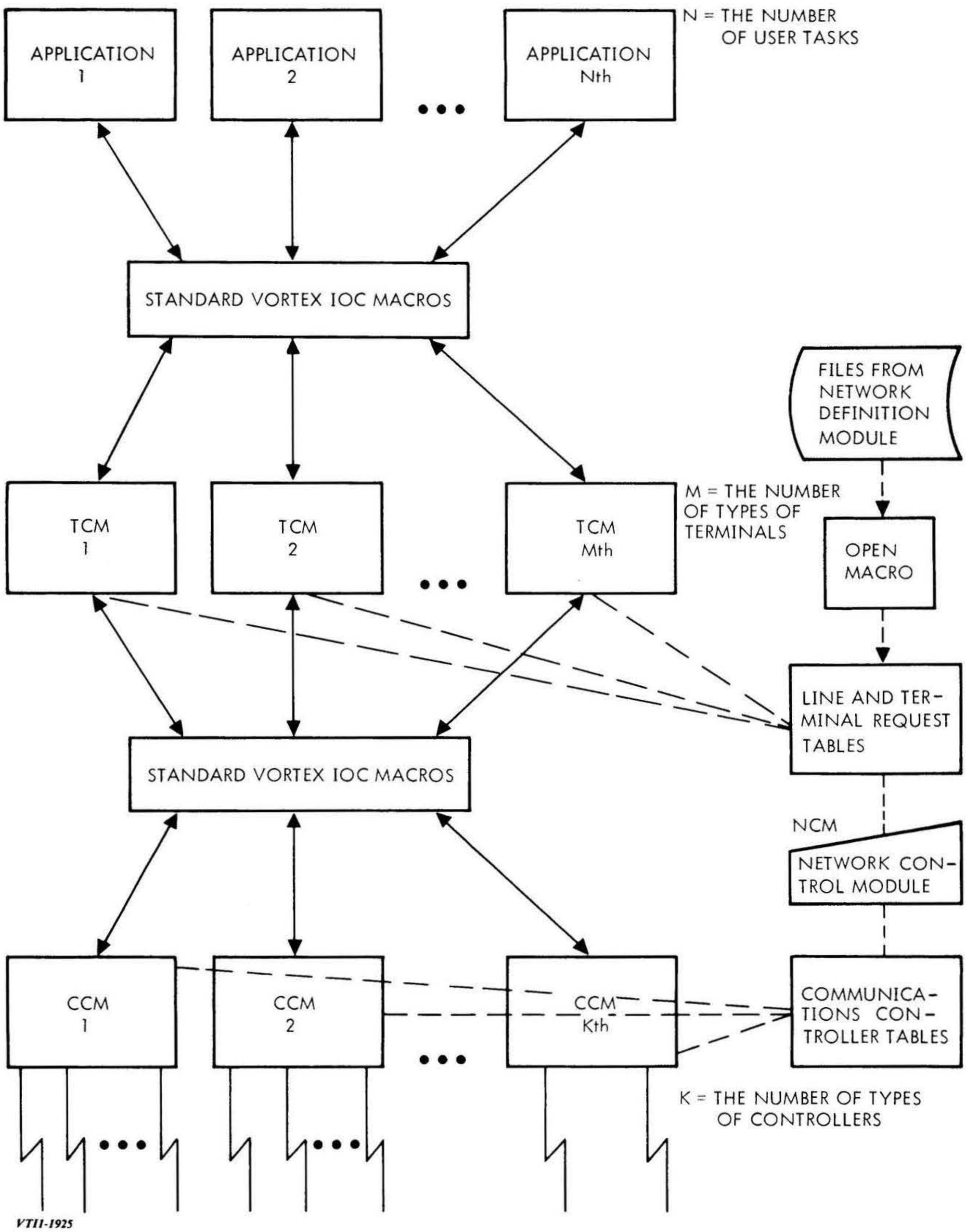
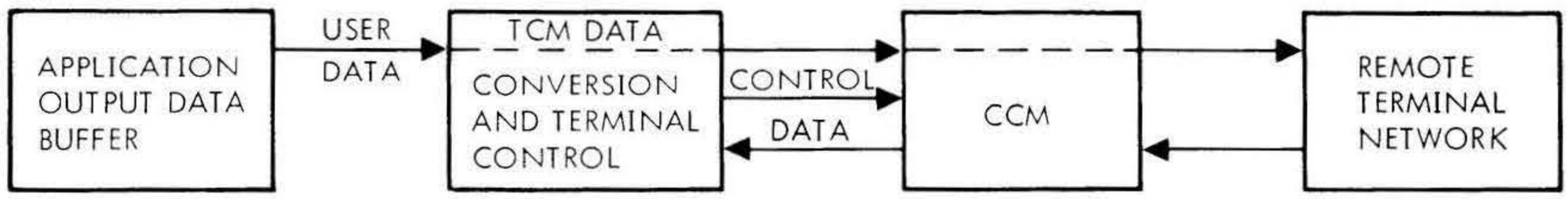


Figure 1-1. Structure of VTAM



VTII-1923

Figure 1-2. Data Flow From Application To Terminal

Another VTAM module aids the user in configuring a network. Because a communications system changes relatively frequently, the method of configuring a VTAM system is less involved than a complete VORTEX system generation. VORTEX SGEN configures the controllers as they are more static than lines and terminals. The VTAM Network Definition Module (NDM) configures the actual terminals, their lines and TCM's. The user determines his line and terminal network and expresses it in the Network Definition Language (NDL). The VTAM NDM interprets the NDL statements and builds the appropriate tables to be used by other VTAM modules (see figure 1-3).

1.3 HARDWARE SUPPORTED AND REQUIRED

The modular organization of VTAM allows its use with a wide variety of configurations depending upon the level at which the user interfaces with the system.

Minimum Configuration

With only the minimum configuration the user must interface with a communications controller module. The following hardware is required.

a. Minimum VORTEX Configuration

b. 52xx Data Communications Multiplexor (DCM) with the proper line adapters or Binary Synchronous Communications facilities.

c. Terminal units which may be supported by the above communications controllers

Expanded Configuration

In addition there may be additional multiplexors to which more terminals are attached where appropriate. Teletype and equivalent terminals compatible with Teletypes can be added.

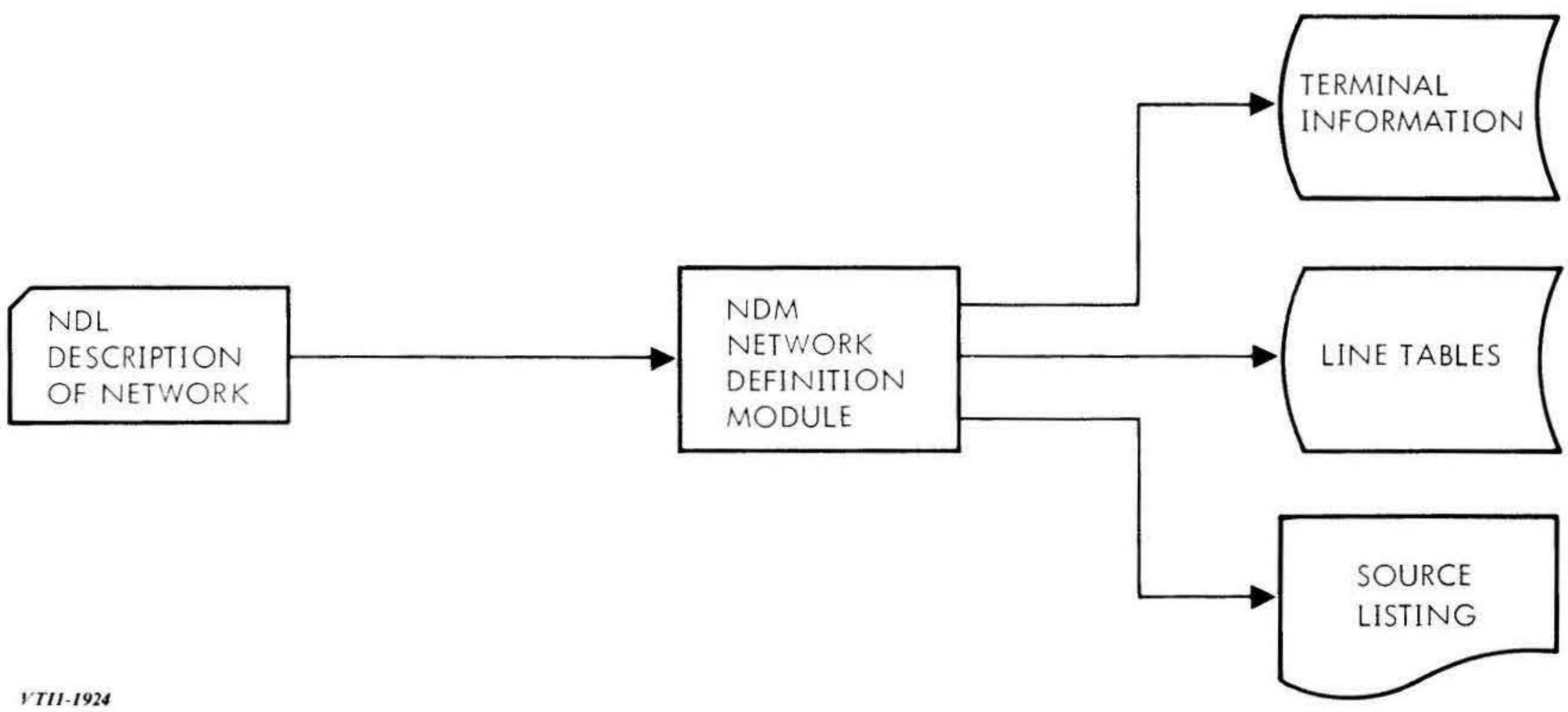
1.4 GUIDE TO THIS MANUAL

This manual explains the VTAM system for a programmer who understands VORTEX, general communication concepts and the computer on which he intends to implement data communications.

The remainder of this section provides a bibliography of related Varian documents.

The remaining sections correspond to components of the VTAM system.

Section 2 describes the Network Definition Language (NDL) and the functions of the module which processes NDL. The next section, 3, provides general information about the macros which the user calls to invoke I/O services of the VTAM modules. Sections 4 and 5 discuss particular macros. Understanding sections 3, 4 and 5 permit an application to communicate with a remote device with limited control and flexibility.



VTII-1924

Figure 1-3. Input and Output To Network Definition Module

Section 6 "Programming at the CCM Level" provides an interface which is more directly involved with the communications lines.

Section 7 "Buffer Chaining" describes the method of automatic buffer chaining on input.

Section 8 "Binary Synchronous Communication" provides information to operate in BSC mode. BSC expands the capabilities of VTAM through its ability to accommodate a variety of transmission codes.

Section 9 "Managing Buffers" describes some macros useful for minimizing the central memory and application uses.

Section 10 "Writing a TCM" provides information which allows adding TCMs for additional types of remote devices.

Section 11 "Controlling a Network" describes the operator interface with a data communication network.

Section 12 is a sample application which illustrates many aspects of the preceding information.

Section 13 describes some additional considerations for a VORTEX system generation on a VTAM system (this information supplements the VORTEX Reference Manual).

Syntax Conventions Used in this Manual

In the directive formats given in this manual:

- **Boldface type** indicates an obligatory parameter.
- *Italic type* indicates an optional parameter.
- Upper case type indicates that the parameter is to be entered exactly as written.
- Lower case type indicates a variable and shows where the user is to enter a legal value for the variable.

A number with a leading zero is octal, one without a leading zero is decimal, and a number in binary is specifically indicated as such.

1.5 BIBLIOGRAPHY

The following Sperry Univac manuals are pertinent to the use of VTAM (the x at the end of each document number is the revision number and can be any digit 0 through 9):

Title	Document Number
VORTEX I Reference Manual	98 A 9952 10x
VORTEX II Reference Manual	98 A 9952 24x
Data Communications Multiplexor Manual	98 A 9902 25x

Additional technical information is contained in the Software Performance Specifications 89A0240 (Overview and External) and 89A0263 (Internal).

SECTION 2
DEFINING A COMMUNICATIONS NETWORK

2.1 INTRODUCTION

The VTAM user describes his terminal and line configuration in the SPERRY UNIVAC Network Definition Language called NDL. The features of the terminals and lines in NDL are processed by the **Network Definition Module (NDM),** which then creates a table of characteristics during input/output request processing (see figure 1-3). This table of characteristics is stored by the processing module on a rotating-memory device (RMD) for expansion and use by other components of the VTAM software in an active network.

The network definition language has three types of statements. These are descriptive rather than procedural. A LINE statement describes the attributes of a communications line. A TERMINAL statement gives the important physical attributes of a remote terminal on a line, and the line to which it is connected. A communications network is defined by these statements for all its terminals and lines followed by an END statement.

Each line is identified by a logical line number and each terminal by a four-character terminal unit identifier. A terminal can only be associated with one logical line number.

2.1.1 Input to the NDM

NDL statements can be input on standard 80-column cards or any other equivalent source input. Only the first 72 characters are processed; 73 through 80 are available for identification and sequencing. Within the first 72 characters the NDL statements are free form, allowing the user to structure his description in columns and with spacing as he finds convenient and meaningful.

2.1.2 General Format

The form of an NDL statement is

keyword *id : attrib(1)=cond(1),attrib(2)=cond(2),..., attrib(n)=cond(n).*

where

keyword	is the word which identifies the statement type such as LINE, TERMINAL or END.
id	is either a logical line number or terminal unit identifier required in line or terminal statements respectively.
each *attrib*	is associated with the particular statement
each *cond*	is associated with the particular attribute

Each descriptive statement must be terminated with a period. Its omission will cause an error indication.

Attributes are optional. For all attributes not specified by the user, NDM assigns default characteristics which are listed in the following sections on particular statements.

2.2 NETWORK DEFINITION LANGUAGE SUBROUTINES and STATEMENTS

The Network Definition Language (NDL) is a FORTRAN mainline program that is activated by the JCP LOAD directive. NDL loads and passes control to the following NDL major subroutines:

Subroutine	Description
CLEAR	CLEAR is called by the NDL mainline routine as an overlay to initialize the two VTAM disc files VT$DFL and VT$DFT, and COMMON storage.
PARSE	PARSE is called by the NDL mainline routine to parse the user's card input. The result of this parse is the complete VTAM files VT$DFL and VT$DFT.
REPORT	REPORT is called by the NDL mainline routine to produce an audit listing of the VTAM files VT$DFL and VT$DFT.

2.2.1 LINE Statement

The LINE statement describes a logical line and its attributes. Upon detecting the initial word LINE, the processor builds a prototype or partial Line Service Descriptor (LSD) for the line and stores it in an RMD file.

The general form of the LINE statement is:

LINE llid: *attrib(1)=cond(1),attrib(2)=cond(2),..., attrib(n)=cond(n).*

Attributes and their corresponding values are as follows:

Attribute	Allowed Values and Meanings	
ADDRESS	nnn	Physical line number 0 through 255
CONNECT	DIRECT D	no modem
	MODEM	non-dial modem
	DIAL-MODEM	dial modem on phone line
EOM-STOP*	FALSE B	message is terminated only when buffer is full or on possible line error
	nnn	specifies the numeric value of the character to terminate input message
	(nnn,nnn) D	specifies (as above) two characters either of which will terminate a input message.
ERROR-STOP	TRUE	terminates input on a line error detected (break, parity or overflow)
	FALSE D	terminates normally on EOM-STOP character, or if EOM-STOP is specified as FALSE, when character count is zero.
PARITY	NONE D	no parity check is to be made
	ODD	odd parity is checked
	EVEN	even parity is checked
STATUS	UP D	the initial state of the line is up
	DOWN	the initial state of the line is down
SPEED	nnn	incoming data rate in characters per second; zero indicates that the data rate is greater than 2000 or less than 4 characters per second.
LINE-TYPE	HALF-DUPLEX	one direction at a time
	SIMPLEX-RECEIVE	one direction all the time only input
	SIMPLEX-TRANSMIT	one direction all the time only output
	FULL-DUPLEX	two way simultaneously
MODE	ASYNCHRONOUS	An asynchronous line, which is described further by attributes following.
	SYNCHRONOUS	synchronous line which is described by additional synchronous attributes.
	BSC	Binary Synchronous Communication line discipline and BSC line adapter use only

*The EOM-STOP attribute is not used for control character detection when in BSC mode.

Attributes only applicable to asynchronous lines. Use of these parameters with synchronous mode is detected and a warning message issued, but the specified action is taken,

Attribute	Allowed Values and Meanings	
ECHO	TRUE	data communications multiplexor operates in ECHO mode for input messages.
	FALSE	no transmission back to terminal of characters received in any input messages
TRANSMIT-SPEED	HIGH = 1	speed of line adapter is set high
	LOW = 0	speed of line adapter is set low

The following six attributes are only applicable to a synchronous line. If the mode is specified as asynchronous the use of these attributes will be flagged and a warning message issued but the specified action will be taken.

Attribute	Allowed Values and Meanings	
CRC-STOP	nnn	the number of characters to be read and stored in the buffer after an EOM character. These characters are not placed in the buffer if it is full. CRC-STOP = 0 disables this function.
STORE-SYNC	TRUE	store any SYNC characters received in buffer
	FALSE	discard any SYNC characters received
SYNCHRONIZE	TRUE	synchronize the line before each receive
	FALSE	do not synchronize line before each receive
SYNC-TRANSMIT	nnn	the numeric value of character sent to the terminal for SYNC
SYNC-RECEIVE	nnn	the numeric value of the character received from the terminal for SYNC.
TRANSPARENT	TRUE	8-bits without parity
	FALSE	7-bits with parity (eight is parity bit)

where **llid** is the logical line identifier which is a number in either octal (with the initial digit a zero) or decimal notation (0 to 254; 255 flags the line as unopened). The attribute list is optionally formed from the line attributes which each have a limited number of conditions to which they can be set. The colon after the logical line identifier and the period at the end of last condition are required.

Only one assignment to a particular attribute may be made. A duplicate will cause processing to continue with the second value replacing the first. Uppercase words indicate those letters are the actual values allowed. Lower case are generic terms.

In general the assignment of an attribute in a line statement may be repeated and causes the last occurrence to override prior settings. For example, if ADDRESS = 012 is specified *after* ADDRESS = 024 the line address will be assigned to address 012.

The following default settings are provided by the network definition module when the attribute is not specified by the user:

Line Attribute Defaults

ADDRESS = 0,
CONNECT = DIRECT,
* EOM-STOP = (0212, 0215),
ERROR-STOP = FALSE,
PARITY = NONE,
STATUS = UP,
SPEED = 0,
LINE-TYPE = HALF-DUPLEX,
MODE = ASYNCHRONOUS,

* = 0212 and 0215 represent the octal values for line feed and carriage return, respectively.

Asynchronous Line Defaults

ECHO = FALSE,
TRANSMIT-SPEED = LOW,

Synchronous Line Defaults

CRC-STOP = 0,
STORE-SYNC = TRUE,
SYNCHRONIZE = FALSE,
SYNC-TRANSMIT = 0226,
SYNC-RECEIVE = 0226,
TRANSPARENT = FALSE,

Examples of LINE Statement

Example 1:

Define a direct-connect line at physical address 012 as logical line number 1 with even parity, incoming data rate of 10 characters per second and messages terminated only when the buffer is full.

LINE 1: ADDRESS = 012, PARITY = EVEN,
SPEED = 10, EOM-STOP = FALSE.

By default the line is direct-connect.

Example 2:

Define a direct-connect line with physical line address 024, as logical line number 2 .

The line has even parity, a data rate of 10 characters per second. Incoming messages are terminated with either a line feed (0212) or carriage return (0215), which are the default EOM characters.

LINE 2: ADDRESS = 024, PARITY = EVEN,
SPEED = 10.

2.2.2 TERMINAL Statement

The TERMINAL statement describes a remote device and declares a set of attributes for it. For each TERMINAL statement the NDM builds a prototype Terminal Control Description (TCD) for the terminal and stores it in an RMD file.

The general form of the TERMINAL statement is

TERMINAL **tuid** : *attrib(1) = cond(1),*
attrib(2) = cond(2),...,attrib(n) = cond(n).

where **tuid** is the unique terminal unit identifier formed from one to four alphanumeric characters. The first characters must be alphabetic A-Z. A duplicate terminal identifier will be flagged and the attributes associated with it will replace those from the prior occurrence.

The terminal attributes that are set in this statement are listed below. Items in upper-case letters are entered as the actual values; lower-case letters represent a position where one type of entry is allowed. For example nnn represents a position for a numeric value either in octal or decimal notation.

Attribute and condition pairs are separated by commas (or equal signs). The list must be terminated with a period.

Attribute	Allowed Values and Meanings	
DEVICES	nnn D=1	specifies the number of devices attached to the terminal
CODE	ASCII D BAUDOT	specifies the code type for the terminal
ECHO	TRUE D	characters inputted are to be transmitted back to the terminal by the TCM (only applicable to a full-duplex line)
	FALSE	no echoing by CCM
LINE	nnn D=0	logical line number to which the terminal is attached
PROMPT	nnn D=0207	numeric value of the character to be sent to terminal when input data is requested
TYPE	TTY1 D	specifies TCM as type 0
	TCMn	(TTY) or n where n is between 1 and 9
UNIT	nnn D=0	logical unit number of the communications controller module
STATUS	UP D	initial terminal status is up (available to be opened)
	DOWN	initial terminal status is down, not available until operator action

The following are the default conditions, provided by the NDM when not specified by the user:

CODE = ASCII
DEVICES = 1
ECHO = TRUE
LINE = 0
PROMPT = 0207
TYPE = TTY1
UNIT = 0
STATUS = UP

The following table shows the net effect of the possible combinations of the ECHO attribute in line and terminal directives:

Attribute	Value			
Line ECHO	TRUE	TRUE	FALSE	FALSE
Terminal ECHO	TRUE*	FALSE	TRUE*	FALSE
Result:	TRUE	TRUE	FALSE	FALSE

*When ECHO is set TRUE concurrent READ and WRITE on a full-duplex line are inhibited.

Examples of TERMINAL Statement

Example 1:

Define a Teletype terminal that is identified as RM01 on logical line number 5. Input characters are not to be echoed back to the terminal.

TERMINAL RM01 : ECHO = FALSE, LINE = 5.

Example 2:

Define a Teletype-compatible terminal that is identified as RM02 on logical line number 6. A carriage return is to be output to the terminal as a prompt character.

TERMINAL RM02: LINE = 6, PROMPT = 0215.

2.2.3 END Statement

The END statement indicates the final entry in the NDM input. It is required and its omission may result in incorrect processing of the description. The only form of this statement is the word END followed by a period.

2.3 OPERATING INSTRUCTIONS

The Network Definition Module of the VTAM system resides in the VORTEX background library. NDM is executed as a background program at priority level 0.

NDM files

Input records to NDM (the NDL statements) are read from the PI logical unit; listings are output to the LO logical unit. The listing includes source language statements, error messages if any occurred, and a summary of characteristics of the network.

The files which contain the tables constructed by NDM are named VT$DFL (for lines) and VT$DFT (for terminals). These files must reside in the FL (foreground library) logical unit.

Example:

Create the required VTAM file and execute NDM.

```
/JOB
/FMAIN
CREATE,FL,F,VT$DFL,120,11
CREATE,FL,F,VT$DFT,120,3
/LOAD,NDM
LINE 1: ADDRESS=012,PARITY=EVEN,SPEED=10.
LINE 2: ADDRESS=024,PARITY=EVEN.
TERMINAL RM01: LINE=1.
TERMINAL RM02: LINE=2.
END.
/ENDJOB
```

The line file VT$DFL is always 11 sectors. The size of the terminal file depends upon the number of terminals. The minimum number of sectors in the file are calculated by integer division as follows:

$$\text{Sectors} = \frac{\text{ntuid} - 1}{29} + \frac{\text{ntuid} - 1}{24} + 2$$

where:

ntuid = number of terminal unit identifiers to be created for the network.

2.4 ERROR INDICATIONS AND WARNINGS

The diagnostic facilities of the NDM produce messages which are warnings and do not terminate processing.

Messages

**ILLEGAL ATTRIBUTE TYPE SPECIFIED

This message indicates a inappropriate value assignments to an attribute. For instance, specification of an asynchronous parameter on a synchronous line.

**SYNTAX ERROR

A syntax error such as a misspelling or an omitted special character (period or colon) followed by the character string where the error is detected.

If the initial word in a statement is not recognized a syntax error message is given and the entire statement to the next period is ignored and processing continues from there.

**DUP TUID NAME

This message indicates more than one terminal statement used the same identifier. The attributes occurring with the latest statement will be assigned.

** FILE VT$DFL TOO SMALL
** FILE VT$DFT TOO SMALL

This message indicates that the named file was not large enough. VT$DFL must be at least eleven sectors. The size of VT$DFT only causes an error message if it is less than two sectors.

I/O Errors

Fatal errors occur as stops with a number indicating which device had an error, EOF or EOD.

Message	Device
NDM STOP 100	PI
NDM STOP 200	LO
NDM STOP 300	VT$DFL
NDM STOP 400	VT$DFT

STOP 100 also occurs on a missing END statement.

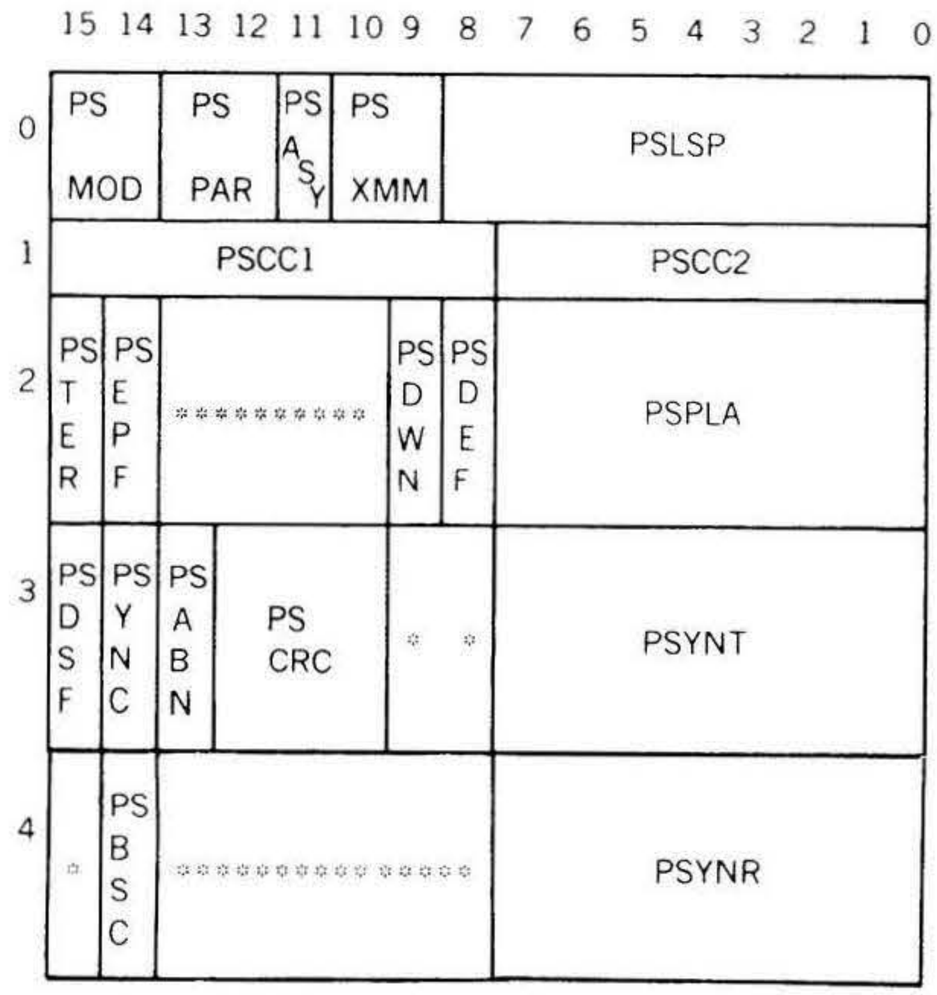
The STOP is given immediately after the I/O operation causing the fault. Thus the last line listed is the card previous to the card causing the fault.

2.5 NDM OUTPUT

As the NDL processor inputs each 80-character record it outputs the record (exactly as input) to the LO unit.

After the END statement is processed the NDM produces a report of the contents of the VTAM files VT$DFL and VT$DFT. The first part of this report lists all defined prototype LSD's in the file VT$DFL. These are listed in order of their logical line numbers. For each defined prototype LSD the logical line number is listed in decimal followed by the five-word descriptor listed in binary (table 2-1 lists descriptions of the prototype LSD fields).

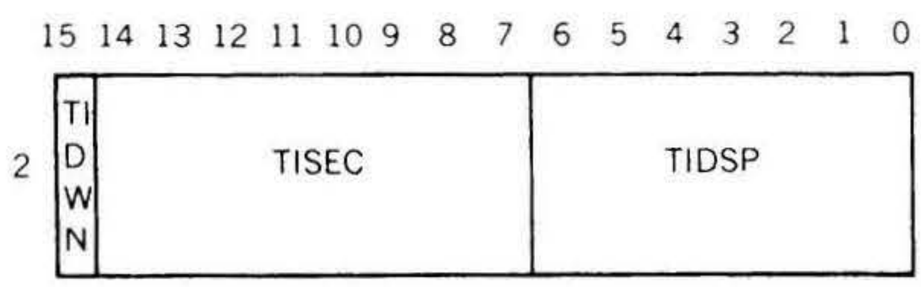
Prototype LSD Output Format



* reserved for future use

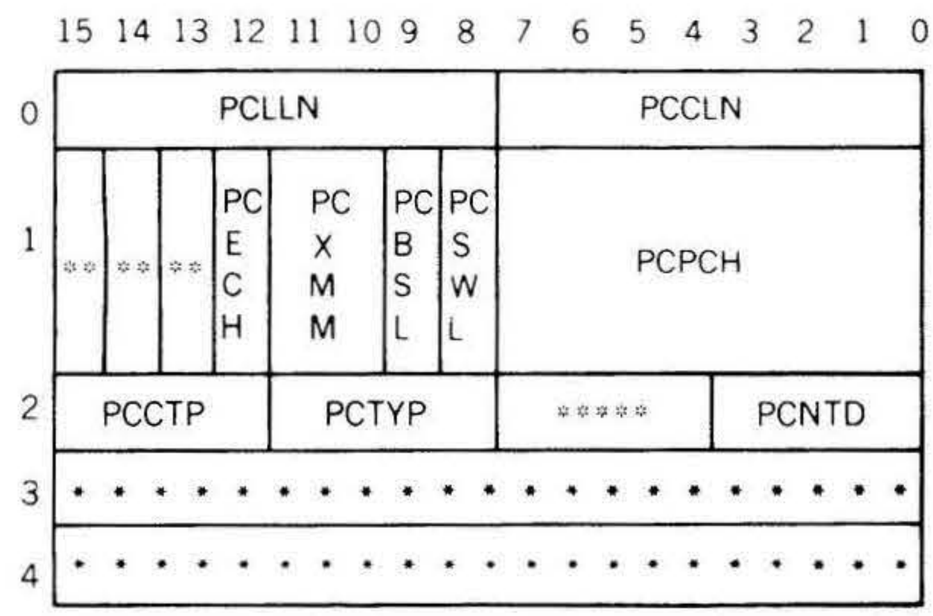
Following the prototype LSD listing the TIB and prototype TCD are listed for each defined terminal. The first line contains the TUID followed by the third word of the TIB in binary.

TIB Output Format



This word indicates the status of the terminal and the RMD location of the prototype LSD. Table 2-2 lists the value and attribute for each field. Next the five words of the prototype TCD are listed in binary. Table 2-3 lists the value and attribute for each field.

Prototype TCD Output Format



* reserved for future use
**not applicable to prototype TCD output

Table 2-1. Prototype LSD Field Description and Range

Field	Attribute	Range
PSMOD	Connection	0 = Direct 1 = Non-dial modem 2 = Dial modem
PSPAR	Parity	0 = No parity 1 = Odd parity 2 = Even parity
PSASY	Mode	0 = Asynchronous 1 = Synchronous
PSXMM	Line-type	0 = Half duplex 1 = Simplex receive 2 = Simplex transmit 3 = Full duplex
PSLSP	Speed	Line speed
PSCC1	Control	Control character 1
PSCC2	Control	Control character 2
PSTER	EOM-stop	0 = False 1 = True
PSEPF	Echo (Asynchronous)	0 = False 1 = True
	Transparent (Synchronous)	0 = 7 bits plus parity 1 = 8 bits (no parity)
PSDWN	Status	0 = Up 1 = Down

(continued)

Table 2-1. Prototype LSD Field Description and Range *(continued)*

Field	Attribute	Range
PSDEF*	Line Status	0 = Line is not defined 1 = Line is defined
PSPLA	Address	0-255
PSDSF	Transmit-speed	0 = Low speed 1 = High speed
PSYNC	Store-sync	0 = Stored 1 = Not stored
PSABN	Error-stop	0 = False 1 = True
PSCRC	CRC-stop	0-7
PSYNT	Sync-transmit	0-255
PSBSC	BSC mode	0 = Not BSC mode 1 = BSC mode
PSYNR	Sync-receive	0-255

*This bit is not set by an attribute; it is set when a line has been defined.

Table 2-2. TIB Field Description and Range

Field	Attribute	Range
TIDWN	Terminal status flag	0 = Up 1 = Down
TISEC	VT$DFT file sector	
TIDSP	VT$DFT file displacement	

Table 2-3. TCD Field Description and Range

Field	Attribute	Range
PCLLN	Line	0-255
PCCLN	Unit	0-255
PCXMM	Line-type	0 = Half duplex 1 = Simplex receive 2 = Simplex transmit 3 = Full duplex
PCBSL	BSC mode	0 = Asynchronous mode 1 = Synchronous or BSC mode
PCSWL	Switched line flag	0 = Up 1 = Down
PCPCH	Prompt	0-255
PCCTP	Code	0 = ASCII
PCTYP	Type	0 = Teletype 1-9 = TCM type 10-15 = Unassigned
PCNTD	Devices	1-15
PCECH	Echo	1 = False 0 = True

The following is an example of the NDM printed output.

```
PAGE    2  03/19/74  NDM        VORTEX  VTAM  NDL

LSD   0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    1 0 0 0 1 0 1 0 1 0 0 0 1 1 0 1
    1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0
    0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0

LSD   1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    1 0 0 0 1 0 1 0 1 0 0 0 1 1 0 1
    1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0
    0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0


PAGE    3  03/19/74  NDM        VORTEX  VTAM  NDL

TIB TTY1
    0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
PCD TTY1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

TIB CRT1
    0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1
PCD CRT1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 /ENDJOB
 /FINI
```

SECTION 3
USING VTAM MACROS

3.1 INTRODUCTION

VTAM requests are written in assembly language as macro calls. The DAS MR assembler provides the following macros for data communications I/O:

OPEN	open a line or terminal
CLOSE	close line or terminal
READ	input from terminal
WRITE	output to terminal
WEOF	write end-of-file designator
FUNC	function request
STAT	status request
LCB	generate a line control block
DCB	generate a data control block

The VORTEX and VTAM systems perform a validity check on all I/O requests. VTAM then queues each valid request to the terminal control module or communications controller module assigned to the specified logical unit. If the appropriate TCM or CCM is not scheduled, the VTAM system schedules it to service the queued requests.

The assembler expands the macros to several words of executable code and data. Certain VTAM operations require parameters in addition to those in the macro call. These parameters are in a table called the line control block (LCB). In general, embedded optional parameters can be omitted by indicating the normal number of commas.

Error messages applicable to these macros are given in section 3.3.

3.2 GENERAL FORM

The general form for data communications I/O macros is:

label **name** **cb,lun,***wait,mode*

where

cb	is the address of a control block
lun	is the logical unit name or number
wait	is the wait/immediate return flag
mode	is the mode of read/write request

The expansion of the macro is generally as follows:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	JSR,X															
1	V$IOC Entry Address															
2	C	Status						E*	CC*		PRIORITY					
3	W	Mode			Op-code				Logical Unit No.							
4	LCB/DCB Address															
5	Actual Number of words or bytes transferred*															
6	Detail Status*															

*only valid when C = 1, (request complete) and not in buffer chain mode.

Explanation of Macro fields

C	Set indicates request complete
Status	Status of I/O request
E	Set if an irrecoverable error occurred
CC	Completion code
priority	Initially zero, set to requesting task's priority by V$IOC
W	Set for immediate return, reset for suspension of calling task until completion of I/O
Mode	Depends upon the particular macro

Op-code specifies the operations to be performed as follows:

0	READ
1	WRITE
2-4	Undefined (return request complete status, if executed)
5	FUNC
6	OPEN
7	CLOSE

LCB/DCB Address	Address of line control block or data control block
Detail Status	the format of the detail status word is shown in table 3-1.

Table 3-1. Detail Status

Bit	Set	Reset
0	Clear to send ON	Clear to send OFF
1	Data set ready ON	Data set ready OFF
2	Ring circuit ON	Ring circuit OFF
3	Carrier ON	Carrier OFF
4	Reverse channel ON	Reverse channel OFF
5	Parity error**	No parity error
6	Control character detected (Read buffer terminated, request complete)	No control character detected
7	I/O clear occurred	No I/O clear
8	Break/format error @	No break or format error
9	Overflow* **	No overflow

* hardware unable to service line before data lost

** In BSC operations, if bit 5 and bit 9 are on, it is an indication of a three second receive timeout (chapter 8).

@ Format error in BSC mode

3.3 ERROR INDICATIONS ON VTAM MACROS

The following I/O errors are given by VTAM in addition to those described in the VORTEX/VORTEX II Reference Manual.

1033	invalid terminal identifier or logical line number
1034	terminal or line not opened
1035	terminal or line down
1036	terminal or line already open
1037	requests still pending
1040	I/O action attempted on terminal not opened
1041	break detected
1042	invalid physical line address
1043	invalid TCM type
1044	no temporary storage available from VTAM memory allocation table
1000	I/O clear occurred
1071	overflow detected

SECTION 4
OPENING AND CLOSING TERMINALS AND LINES

4.1 INTRODUCTION

When an application program requires the services of VTAM to communicate with a remote device both the line and terminal must be opened before any I/O action. A READ or WRITE on an unopened line or terminal will result in an error message. An OPEN request for a terminal will also open the line on which that terminal is attached. Lines can be opened but must be followed by a terminal OPEN request in order to establish communication with the terminal.

Line Open and Close Actions

On a line open request the prototype Line Service Descriptor (LSD) is read from the RMD file VT$DFL. A block of temporary storage is obtained from a memory pool, and a central memory resident LSD is built. The LSD is linked through the physical line table to the logical line table and also to the LSD queue. This procedure makes the line available to the user to make I/O requests.

On a line close, the LSD is removed from the physical line table and also from the LSD queue. The temporary storage block is returned to the memory pool and is available to another user.

Terminal Open and Close Actions

When an application requests an open or close on a terminal, V$IOC passes the request to the appropriate TCM, which functions as a pseudo driver under VORTEX.

The terminal open request causes the prototype terminal controller descriptor (PCD) to be read from the file VT$DFT. If the line for this terminal is not already open, an open request is made for the line. Upon return a block of temporary storage is obtained from a memory pool and a TCD resident in main memory is built. The TCD is linked to the logical terminal table and also to the TCD queue. After successfully completing this procedure the terminal is available for the user to make READ, WRITE, CLOSE etc. requests.

The terminal close request removes the TCD from the logical terminal table and from the TCD queue. The temporary storage block is returned to the memory pool, thus making the area available to another user. If there are no more terminals open on the line, a close request is made to close the line.

Open and close requests are coded in the applications software as macro calls in DAS MR, as subroutine calls in FORTRAN and also as JCP directives.

JCP Open and Close Actions

To provide the ability to perform line and terminal opening and closing external to an application program, JCP allows these actions through the OPEN and CLOSE directives. In effect lines and terminals may be opened and closed either through macros within a program or through the JCP directives /OPEN and /CLOSE before or after program execution. These directives also allow system I/O units like LO to be opened and assigned to a line or terminal by the second format of OPEN (see section 4.2.1). In this manner a user or the system operator has the option of opening and closing lines or terminals outside of a program by entering these JCP directives through the SI device.

4.2 OPEN MACRO AND JCP DIRECTIVE

The OPEN macro is applicable to either a line or terminal.

4.2.1 Forms of OPEN Macro

DAS MR:

label **OPEN** **cb,lun,***wait*

where

cb	is the address of the line control block (LCB) or the data control block (DCB) containing the four-character terminal unit identifier in the first two words.
lun	is the logical unit number for the CCM opening a line or the TCM opening a terminal
wait	is 1 for an immediate return or 0 (default) for suspension of the caller until the open is complete

FORTRAN:

label **CALL** **VT$OPN (name,lun,stat)**

where

name	is a three-word array containing the LCB or DCB
lun	is the logical unit number for the CCM opening a line or TCM opening a terminal
stat	is an integer variable where the status will be returned

All FORTRAN open requests cause suspension of the calling program until the open is complete.

JCP:

/OPEN, lun_1, id

/OPEN, lun_2, id, lun_1

where

lun_1	is the logical unit name or number for the CCM opening a line or TCM opening a terminal
id	is the logical line number for opening a line or terminal unit identifier for opening a terminal
lun_2	is the logical unit name or number which will be assigned to the CCM or TCM designated by the other lun after the terminal has been opened

Example :

```
/OPEN,184,TTY1
/OPEN,LO,TTY2,184
```

Note: A JCP /OPEN directive performs two functions: a VTAM open and a VORTEX IOC assignment. Therefore, a VORTEX job stack containing a /OPEN directive should also contain a /CLOSE directive before any JCP reassignment directives (such as /FINI or /JOB) are input. If, for example, a /OPEN is followed by a /FINI, the VORTEX IOC linkage to VTAM is broken, but the VTAM linkage remains intact. To recover from this, use a JCP or OPCOM ASSIGN directive.

Example:

Suppose a TCM has device name TC00 and logical unit number 184. An NDL directive defines a terminal name TR15. A VORTEX job wants to direct LO output to this terminal, thus:

/OPEN,LO,TR15,184
/FINI

The /FINI is in error, for it will reassing LO back from terminal TR15 to its default value, but VTAM tables and linkage are left in an incorrect state. To recover, enter:

/ASSIGN,LO,TC00

and all linkage is reestablished.

4.2.2 Error Indications on OPEN

DAS MR:

The open/close module generates the following status in word two of the request, bits 14-5 for DAS MR OPEN calls:

Bit	Value	Meaning (Standard VORTEX error message codes)
14-9	00	normal completion
	02	invalid lun for CCM or TCM
	033	invalid logical line number or tuid
	035	line or terminal down
	036	line or terminal already open
	042	invalid physical line address
	043	invalid TCM type
	044	no temporary storage available for LSD or TCD
8	1	irrecoverable I/O error
7-5	0	normal return
	5	I/O error

FORTRAN:

The open/close module returns the following status as a result of a FORTRAN OPEN call:

Contents of STATUS	Meaning
0	normal completion
1	invalid lun for CCM or TCM
2	invalid logical line number or tuid
3	line or terminal down
4	line or terminal already open
5	invalid physical line address
6	invalid TCM type
7	no temporary storage available
8	I/O errors

JCP:

Any errors as a result of an /OPEN directive to the JCP will result in the error message " JC06" being output to the SO and LO logical units.

Examples of OPEN

Example 1:

Open line 16 on logical unit 72. Select the wait option. The LCB address is TTYLCB.

```
OPEN TTYLCB,72
```

The default value for wait is used. The line number is in the LCB.

The same request in FORTRAN would be:

```
INTEGER TTYLCB, STATUS
DIMENSION TTYLCB( 3 )
TTYLCB(3) = 16
CALL VT$OPN (TTYLCB,72,STATUS)
```

Example 2:

Open a terminal whose tuid is XY03 on logical unit 122. Select immediate return.

```
TUIDCB DCB    'XY''03'
       OPEN   TUIDCB,122,1
```

The same request in FORTRAN (except for the wait for completion instead of immediate return) would be:

```
INTEGER TUIDCB, STATUS
DIMENSION TUIDCB (3)
DATA TUIDCB (1),TUIDCB(2) /2HXY,2H03/
CALL VT$OPN (TUIDCB,122,STATUS)
```

4.3 CLOSE MACRO AND JCP DIRECTIVE

The CLOSE macro is applicable to both lines and terminals.

4.3.1 General Format

DAS MR: for

label **CLOSE** **cb,lun,***wait*

The parameters are identical to those described for OPEN. This is the standard VORTEX CLOSE macro.

FORTRAN:

label **CALL** **VT$CLS(name,lun,stat)**

where **name** is the three-word array containing the LCB or DCB, and **stat** is an integer variable where the status will be returned. All FORTRAN CLOSE requests cause suspension of the calling task until the I/O is complete.

JCP:

/CLOSE,lun,id

where **id** is either the logical line number or the four-character terminal unit identifier, used to open the line.

All JCP CLOSE directives cause suspension of the JCP unit until the CLOSE is complete.

Example:

/CLOSE,LO,TTY2

4.3.2 Error Indications

DAS MR:

The open/close module generates the following status indication in the second word of the request, bits 14-5 for DAS MR CLOSE calls:

Bit No.	Value (Octal)	Meaning
14-9		Standard VORTEX error message code
	00	normal completion
	02	invalid LUN for CCM
	33	invalid logical line number or tuid
	34	line or terminal not open
	37	requests still pending on line or terminal
	43	invalid TCM type
7-5	0	normal return
	5	I/O error

FORTRAN:

The Open/Close module returns the following status as the result of a FORTRAN CLOSE call:

Contents of Status Word	Meaning
0	normal completion
1	invalid LUN for CCM
2	invalid logical line number or tuid
3	line or terminal not open
4	requests still pending on line or terminal
5	invalid TCM type
6	I/O error

JCP:

Any error conditions as the result of a /CLOSE directive to the JCP will result in the error message " JC06" being output to the SO and LO logical units.

Examples of CLOSE

Example 1:

Close previously opened line 16 on logical unit 72. Select the wait option. The LCB address is TTYLCB.

```
CLOSE TTYLCB,72
```

The default values for wait is used. The line number is in the LCB.

The same request in FORTRAN would be:

```
INTEGER TTYLCB, STATUS
DIMENSION TTYLCB (3)
      .
      .
      .
TTYLCB (3) = 16
```

```
CALL VT$CLS (TTYLCB, 72, STATUS)
      .
      .
      .
```

Example 2:

Close a previously OPENed terminal with tuid of ZZ15 on logical unit 201. Select immediate return.

```
TUIDCB DCB      'ZZ','15'
         .
         .
         .
       CLOSE    TUIDCB,201,1
```

The same request in FORTRAN (except for an automatic wait instead of immediate return) would be:

```
INTEGER TUIDCB, STATUS
DIMENSION TUIDCB (3)
DATA TUIDCB (1), TUIDCB (2)/2HZZ, 2H15/
      .
      .
      .
CALL VT$CLS (TUIDCB, 201, STATUS)
```

SECTION 5
PROGRAMMING AT TCM LEVEL

A data communications application program can converse with a remote device through the TCM for that type of terminal. This section describes the use of a standard TCM called TTY for Teletype and similar compatible terminals.

The TTY TCM processes READ, WRITE, FUNC, STAT, and WEOF requests from application programs written in DAS MR and FORTRAN running under VORTEX. These functions can be performed only after the terminal is opened (open actions are described in section 4).

Use of paper tape with VTAM must be consistent with the terminal being used. For instance, a strictly binary data stream transmitted to a Teletype Model 35 ASR could contain the ASCII bit patterns to start and stop the paper-tape punch and reader thus causing loss of information on the resulting paper tape. Similarly, use of the paper-tape reader must be carefully considered because of the absence of control in data being read. Depending upon the processing load on the CPU, one or more data bytes might be lost between logical reads.

5.1 MACRO DEFINITION

All calls to the TTY TCM are processed through the normal IOC component (described in VORTEX Reference Manual). The TCM processes Teletype keyboard input and printer output requests as well as Teletype paper-tape reader and punch operations. The TTY TCM performs READ, WRITE, FUNC, STAT, and WEOF functions but all other IOC macro functions are ignored by the TCM, and are unconditionally returned as I/O complete.

5.2.1 READ Macro

The READ macro operates in two modes, either in standard ASCII or in a transparent mode which does not recognize and react to editing characters and does not perform user prompting or carriage control.

ASCII READ

An ASCII READ request inputs through the TCM from the device one record of up to 80 ASCII characters, or 40 words.

A record is terminated by either a carriage return character or input of the 80th ASCII character. In the latter case a carriage return and line feed are output to the TTY. If a carriage return character terminated the READ, the remaining unused portion of the input buffer is cleared to ASCII blank characters and a line feed is output.

Any input request causes the prompt character such as the BELL character to be output to indicate that the keyboard is ready for input. All valid ASCII characters are stored two characters per word left justified in the user buffer specified in the DCB. All characters are echoed if the terminal is on a full-duplex line and ECHO is set; on a half-duplex line, characters are not echoed but printed locally by keyboard action.

The backslash character (shift and L simultaneously) is a control character to delete the current record. A carriage return and line feed are output to inform the user that a new record can be input.

The backarrow character (shift and the letter O simultaneously) on input deletes the preceding character input. Characters cannot be deleted beyond the current line.

The carriage return character causes the current record to be terminated and the system responds with a line feed. The carriage return is not stored in the user's buffer.

When the ASCII mode is used, the READ request has a timeout feature which is described with the FUNC macro (see section 5.2.4).

The BELL character also has a special function when it is the first character input in response to a READ in the standard ASCII mode. It causes the READ to be terminated and returns end-of-file (EOF) status with the completed READ request. To distinguish this condition from data-set-ready OFF condition (completion code = 6), the irrecoverable error flag is set for the data-set-ready OFF case.

Transparent Mode

This mode is identical to the ASCII mode described above except in the cases listed below.

a. The buffer length specified in the DCB is not limited to 40 words. If the length is greater than 80 characters, the TCM will continue input until a carriage return is received or the buffer is full.

b. A line feed is not output, when the READ is terminated.

c. The unused portion of the buffer is not set to blanks.

d. No prompting character is output.

e. No input editing is performed.

READ Macro

label **READ** **dcb,lun,***wait,mode*

where

dcb	address of the DCB
lun	logical unit number of the terminal
wait	set for immediate return, otherwise program is suspended until I/O complete (0 is the default)

mode mode of read
1 = ASCII (default)
4 = transparent
all other modes reserved for future use and are defaulted to 1

Example of a READ Macro

DAS MR:

Read a record on logical unit 64. Select immediate return option and mode 1.

```
TYUN    EQU     64      (LUN assigned to
        .               terminal via OPEN)
IM      EQU     1       (Immediate return)
STMD    EQU     1       (Standard, ASCII mode)
RECL    EQU     40      (Record length 40
        .               words)
        OPEN    TUID,TYUN
        .
TYRD    READ    TTY,TYUN,IM,STMD
        .
        .
TUID    DCB     'TY','C1'
TTY     DCB     RECL,BUFF (Data control
        .                 block: user data
        .                 area specifying record
        .                 length in words. To
        .                 specify byte count,
        .                 use indirect address
        .                 constant: (BUFF)*)
BUFF    BSS     40        (user data area)
```

FORTRAN:

Read a 20 character record on logical unit 64 into a buffer, packing two characters per word.

```
        DIMENSION IBUFF(10)
        .
        .
        .
        READ(64,100) IBUFF
100     FORMAT(10A2)
```

Return conditions for READ

The TTY TCM generates the following status in the request, word 2 of bits 14-5:

Bit Number	Value	Meaning
14-9		Two octal digits error message code (see VORTEX Reference Manual)
	00	Normal completion or I/O clear
	01	Device declared down
	02	Illegal opcode or unassigned logical unit number
	30	Parity error occurred during data transmission
	40	Terminal not open
	41	Break detected
	71	Overflow detected
8	1	Irrecoverable error
7-5		Completion code
	0	Normal return
	5	Error
	6	End-of-file (Bit 8 = 0) Data-set-ready off (Bit 8 = 1)
	7	Read time-out

5.2.2 WRITE Macro

The WRITE macro like the READ macro operates in two modes, either in standard ASCII or in a transparent mode which does not recognize and react to editing characters nor perform user prompting or carriage control.

ASCII mode (1):

The write request causes the TTY TCM to output one record of ASCII character data of up to 36 words (72 ASCII characters) in length. The record size (in words or bytes) is specified by the user in the DCB. All trailing characters in the specified buffer must be ASCII blank characters. The TCM determines the actual number of characters to output by starting at the end of the buffer and counting the number of trailing ASCII blank characters, then subtracting this count from the maximum number of characters possible in the buffer.

When a record is output to the Teletype printer, the first character of the record is reserved for a vertical spacing character and is not printed. The TCM will replace the first character with a blank character. The vertical spacing control characters have the following meaning:

ASCII Character	Vertical Spacing
Blank	One line (single space)
0	Two lines (double space)
1	ASCII form character is output

When the last character of the buffer has been printed, the TCM outputs the carriage return, null, and line feed characters. The normal completion status is stored in the request block and control is returned to the user if the WAIT option was used.

Transparent Mode (4):

This mode is identical to mode = 1 except as follows:

a. First character in user buffer is not used for forms control.

b. Each character in the buffer is output with no special checking. If more than 72 characters are output on one line, no action is taken by the TCM.

c. All forms control is handled by characters in the user's buffer. Upon completion of printing the user's buffer, no carriage return, null, and line feed characters are output.

The format of the WRITE macro is:

label **WRITE** **dcb,lun,***wait,mode*

Where the parameters are the same as defined for the READ macro.

The TTY TCM generates the following status in the request, word 2 of bits 14-5:

	Bit Number	Value	Meaning
STATUS	14-9		Two octal digits for error message code
		00	Normal completion or I/O clear
		01	Device declared down
		02	Illegal opcode or unassigned logical unit number
		40	Terminal not open
		41	Break detected
E	8	1	Irrecoverable error
CC	7-5		Completion code
		0	Normal return
		5	Error
		6	Data-set-ready OFF

Example of a WRITE Macro

DAS MR:

Write a record on logical terminal 64. Select the wait option and mode 4.

```
TYUN    EQU     64   (LUN assigned to
        .            terminal via OPEN)
WAIT    EQU     0    (Wait option)
WRMD    EQU     4    (Transparent mode)
RECL    EQU     120  (Record length 120
        .            bytes)
        .
        .
TYWR    WRITE   TTY, TYUN, WAIT, WRMD
        .
        .
        .
TTY     DCB     RECL,(BUFF)* (User
        .       data area specifying
        .       record length in bytes. To
        .       specify word count, use
        .       direct address constant:
        .       BUFF)
BUFF    BSS     60   (user data area)
```

FORTRAN:

Write a 20 character record on logical unit 64 from a buffer, packing two characters per word.

```
        DIMENSION IBUFF(10)
        .
        .
        .
        WRITE(64,100) IBUFF
100     FORMAT(10A2)
```

5.2.3 STAT Macro

The status request macro STAT causes the status to be examined and control transferred to a user-defined routine for the processing of errors.

The format of the STAT macro is:

label **STAT req,err,aaa,bbb, busy**

where **req** is the address of the I/O macro, **err** is the address of the I/O error routine, **aaa** is the address of the data-set-ready OFF routine, **bbb** is the address of the READ request time-out routine, **busy** is the address of incomplete-I/O routine.

Except *label* all parameters are mandatory.

The contents of the overflow indicator and the A and B registers are saved.

Return Conditions

Upon normal completion, control is transferred to the task after the end of this macro expansion.

If an I/O error occurred, control is transferred to the address specified as **err.** If the data-set-ready signal is off, control is transferred to the address **aaa.** If the length of time for a terminal response exceeds the time-out specified in a FUNC macro, control passes to the address **bbb.** An incomplete I/O causes transfer to the address specified as **busy.**

Example of a STAT Macro

Read a record on unit 64 and check for Data-Set-Ready OFF and time-out. Use immediate return option, mode 1.

```
TYUN     EQU     64  (Logical terminal unit)
IM       EQU     1   (Immediate return)
RDMD     EQU     1   (Standard mode)
RECL     EQU     80  (record length)
         .
         .
         .
TYRD     READ     TTY, TYUN, IM, RDMD
         .
         .
         .
B        STAT     TYRD, ERR, DSRO, RTO, B
         .
         .
DSRO              (DATA SET OFF ROUTINE)
         .
         .
RTO               (TERMINAL TIME-OUT
         .           ROUTINE)
         .
         .
ERR               (ERROR ROUTINE)
         .
         .
TTY      DCB      RECL, BUFF (Data
         .        Control Block)
BUFF     BSS      80
```

5.2.4 FUNC Macro

The FUNC request causes the TTY TCM to perform specific functions that cannot be performed by other macros. The value of the low-order bits of the function code word of the DCB defines the operation to be performed.

Function	Function Code	Comments
Output carriage return and 3 line feed characters.	0	Outputs the sequence of characters, sets normal completion status in the request block and control returns to user.
Set NO ECHO flag for READ requests on full duplex lines. This flag is initially reset when terminal is opened.	1	Causes input characters for subsequent READ requests not to be echoed if terminal is on full duplex line.
Reset NO ECHO flag for READ requests on full duplex lines.	2	Causes input characters for subsequent READ requests to be echoed if terminal is on full duplex line.
Set a timeout value for READ requests which use the WAIT option. (Only for the ASCII READ mode.)	3	Sets a timeout value for all subsequent READ requests on the terminal. The default timeout value is zero and this prevents the TCM from performing timeouts for READ requests on the terminal. When this function request is used, the high-order byte of the function code word of the DCB will be used for a timeout value (1-511 secs.) for all subsequent READ requests until it is reset to another value. When a non-zero timeout value has been specified, the TCM will check for a READ timed-out condition while waiting for input. If timeout occurs, timeout status is returned to the user and the number of words/bytes input set to zero in the request block. The TCM also outputs the carriage return (CR) and line feed (LF) characters if mode of request is 1.

(continued)

Function	Function Code	Comments
Set a terminal DOWN and clear all active and pending TCM I/O requests on a terminal	4	This function is used to set an opened terminal DOWN and to clear all active and pending TCM I/O requests on the terminal. The device-declared-down error status is returned for all TCM requests and any CCM I/O requests are cleared. Memory used for CCM request blocks are released. This function is an immediate function. Therefore, it is not queued. Normal completion status is then returned to the user for the function request after the I/O clear has been performed.
Clear READ request.	5	Causes artificial termination of the current active read request (which terminates with CC = 101, E = 1, and STATUS = 00). The request is marked complete as soon as the TCM services the request.
Clear WRITE request.	6	Causes artifical termination of the current active write request. Completion status and timing information is the same as for function code 5.

All other function codes are reserved for future use.

label **FUNC** **dcb, lun,** *wait*

Parameters are the same as described for READ request, except the last word of the DCB, function code word, is used by FUNC requests: function code (bits 7-0), and READ timeout value (bits 15-8) when function code is equal to 3.

Return Conditions:

Return conditions are the same as described for WRITE requests.

Example of a FUNC Macro

Set time-out value of 511 seconds for READ requests on logical terminal unit 64.

```
TYUN    EQU     64              (Logical terminal
                                 unit 64)
        .
        .
        .
        FUNC    TODCB,TYUN      (Set read
        .                        timeout value)
        .
        .
TODCB   DCB     RECL,BUFF,0177403
                (Timeout value = 511
                 seconds, function code = 3)
```

5.2.5 WEOF Macro

The WEOF request causes the TTY TCM to output the terminal prompting character. It indicates to the user that the end-of-file has been reached. The normal completion status is returned in word 2 of the request and control is returned to the user if the WAIT option was used.

General form:

label **WEOF** **dcb,lun,***wait*

The parameters are the same as described for the READ request, though the DCB address is not used by the WEOF request.

Return conditions are the same as for WRITE request (section 5.2.2).

Example of a WEOF Macro

Output user prompting character on logical terminal unit 64. Use immediate return option.

```
TYUN    EQU     64              (Logical terminal
        .                        unit 64)
IM      EQU     1               (Immediate return)
        .
```

```
        .
        .
PROMPT  WEOF    WDCB,TYUN,IM
        .
        .
        .
WDCB    DCB     RECL,BUFF
```

5.3 TTY TCM WITH DIAL-UP LINES

Before any I/O operations can be performed on a terminal, it must have been opened with an OPEN request (section 3.2). If a terminal is defined as on a dial-up line, the action of opening a terminal causes Data-Terminal-Ready to be turned on, to enable answering the ring on the line. When any TCM I/O request is made on a terminal, a check is made for data-set-ready on. If data-set-ready is on, a physical connection flag is set in the Terminal Controller Descriptor (TCD) for the terminal and the request is initiated. If it is OFF, the request is not initiated and remains queued until Data-Set-Ready is ON.

If the physical connection flag has been set and the Data-Set-Ready is off, the TCM considers it a line disconnect and returns Data-Set-Ready OFF as a status to any active or pending TCM requests. In this situation the terminal should be closed and reopened to permit the user to dial up again and get physical connection to the terminal.

5.4 FORTRAN LEVEL PROGRAMMING

Programming at the FORTRAN level follows the normal rules for using FORTRAN READ and WRITE statements. The only additional requirement is that the line be opened and closed using the OPEN and CLOSE macros (see section 4).

use for BSC

SECTION 6
PROGRAMMING AT THE CCM LEVEL

6.1 INTRODUCTION

The CCM functions as a driver for data communications equipment at the communications multiplexor and line level. It processes requests made by terminal control modules or application programs which require a more direct interface with the communication lines than that provided through the TCM.

Line disciplines and modem characteristics are defined in the line-oriented tables of line service descriptors, thus, the user of a CCM need not define these items himself in an application program. Some portions of these tables can be modified dynamically by user programs.

Binary Synchronous Communications (BSC), both half- and full-duplex lines, as well as input in the buffer chaining mode are accommodated by the CCM.

The CCM provides orderly line turnaround in half-duplex operation and permits concurrent READs and WRITEs on full-duplex lines.

6.2 CCM I/O CONTROL MACROS AND FUNCTIONS

The CCM I/O requests are written in assembly language with the following I/O macros.

Name	Function
LCB	Generate a Line Control Block
OPEN	Open a line
CLOSE	Close a line
READ	Read a record
WRITE	Write a record
FUNC	Function request
STAT	Status request

The general form of data communications I/O macros (section 3.2) is also applicable to CCM macros.

6.2.1 LCB Macro

This macro generates a line control block which is required by all data communications I/O requests. The form of the Line Control Block macro is:

label **LCB** **rl,buf,line,***func,c,e*

where

rl is the length in words or bytes of the record to be transmitted or received, the maximum record length is 4096 bytes or 2048 words.

buf is the address of the first word of the buffer.

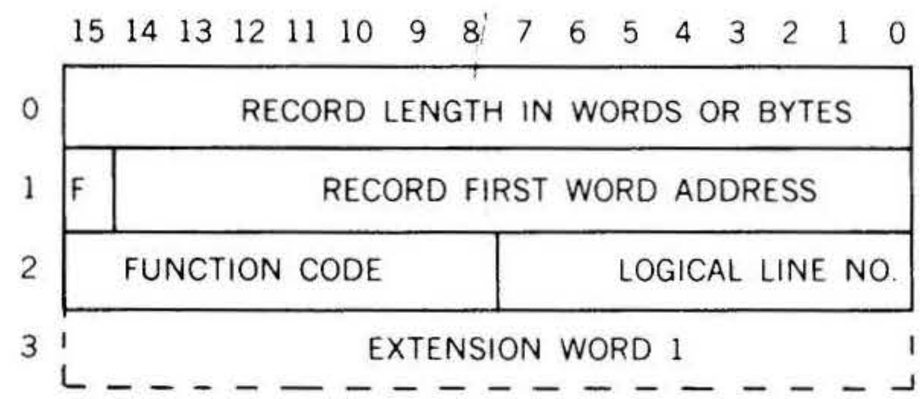
line is the logical line number.

func function code only applicable to FUNC request

c 1, if length is expressed in bytes
0, if length is expressed in words (default value)

e extension, meaning depends upon the function being performed

LCB Macro Expansion is described below.



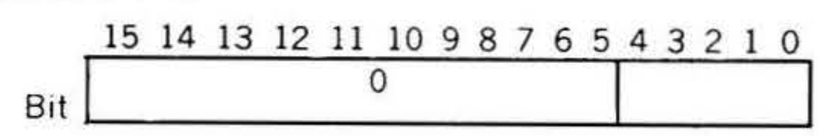
F = 1, record length expressed in bytes. F = 0, record length expressed in words.

Note: If in buffer chaining mode, F must = 1.

FUNCTION CODE = 0-255
LOGICAL LINE NUMBER = 0-255
Optional EXTENSION WORD 1 is used for FUNC requests and for the chain header address in buffer chaining.

Function code 3, sense event.

LCB MACRO + 3



0 - wait for Clear to Send ON/OFF 001

1 - wait for Data Set Ready ON/OFF 002

2 - wait for Ring Circuit ON/OFF 004

3 - wait for Carrier ON/OFF 010

4 - wait for Reverse Channel ON/OFF 020

When the specified event occurs (status changes), FUNC is flagged complete.

Function code 7, load control characters.

LCB MACRO + 3

15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Control Character 1	Control Character 2

Function code 10, load sync characters.

LCB MACRO + 3

15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Transmit Sync Character	Receive Sync Character

Function code 25 or any buffer chain mode read.

LCB MACRO + 3

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Chain Header Address

Function code 6, transmit break.

LCB MACRO + 0

15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
0	rl

rl is the number of character times that the break condition will be maintained on the line, depending on F (LCB word 1).

Examples of an LCB Macro

Define an LCB for a data buffer COMBUF which is 375 bytes in length. I/O operation will be performed on line 37.

```
LCB1        LCB        375,COMBUF,37,1
```

Define an LCB for a data buffer of 20 bytes long, starting at BUF1, to be used for READ in buffer chain mode, in logical line 0, and buffer chain header (CHR) is at INCHR.

```
            LCB        20,BUF1,0,0,1,INCHR
```

Define an LCB for use with a FUNC request. The function code is 3 (sense event) which requires an optional extension word for the event flags. The function will be performed on line 3.

```
LCBC        LCB        0,0,3,3,,014
```

The event word is an octal value of 014 which selects notification when carrier-on or carrier-off and ring-on or ring-off occurs.

Status

Status information is returned to the requesting program in three different fields within the request.

a. e field (word 2, bit position 8) is set when an irrecoverable error has occurred.

b. cc field (word 2, bit position 5-7) is set for use by the STAT request. Interpretation of the values is as follows:

cc bits	Meaning
7 6 5	
0 0 0	normal return
1 0 1	error
1 1 0	reverse channel on, ring detected
1 1 1	asynchronous line break detected

c. Detail status (word 6, bit position 0-15). See Data Communications I/O Macros section 3, table 3-1.

Normal Termination Status

e = 0
cc = 0, 6, or 7
Detail status bit 7, control character detected may be set for a READ request.

Abnormal Termination Status

e = 1
cc = 5
Detail status bits set as follows (see table 3-1):

Bit Number	Value	Meaning
1	0	Error condition if occurred during READ or WRITE.
3	0	Error condition if occurred during READ or WRITE.
5,8,9	1	Error condition during READ only.
7	1	Error is a result of a user generated I/O clear by means of FUNC request.

d. IOC status field in word 2, bits 9-14 of the request. This field is used by IOC and VTAM to notify the requesting program of error conditions relating to the validity of the request (see VORTEX Reference Manual).

6.2.2 OPEN Macro

The OPEN macro is executed to place a line in the active state and then permit I/O requests. I/O requests issued before the line is OPENed will result in an error status return. The CCM OPEN processor establishes the terminal table environment and performs the necessary line initialization. A second OPEN macro returns error status 036 with E = 1 (irrecoverable).

Open Line Macro:

label **OPEN** **lcb,lun,***wait,mode*

where

lcb	is the address of the line control block
lun	is the number of the logical unit used to reference the CCM.
wait	is 1 for immediate return or 0 (default value) for suspension of the caller until the I/O is complete
mode	0 = default value (reserved for future use)

Example of an OPEN Macro

Open line 16 on logical unit 72. Select the wait option and mode 0. The LCB address is TTYLCB.

```
CCMLUN   EQU      72
TTYLJN   EQU      16
         .
         .
         .
TYOPEN   OPEN     TTYLCB,CCMLUN
         .
         .
         .
TTYLCB   LCB      0,0,TTYLJN
```

Wait and mode take default values.

6.2.3 CLOSE Macro

The CLOSE macro is executed to release a line from active use. The CCM CLOSE processor releases table space for the description of the line environment and terminates the hardware and/or software scanning of the line. The form of the CLOSE macro is:

label **CLOSE** **lcb,lun,***wait,mode*

where the parameters are the same as defined for the OPEN macro.

Example of a CLOSE Macro

Close previously opened line 0 on logical unit 107. Select no wait and mode zero. The LCB address is LCB 107.

```
LUN      EQU      107
LJNEN0   EQU      0
WAIT     EQU      1
         .
         .
         .
CLMAC    CLOSE    LCB107,LUN,WAIT
         .
         .
LCB107   LCB      0,0,LJNEN0
```

6.2.4 READ Macro

The read macro causes the CCM to input a data block of a specified length and format.

label **READ** **lcb,lun,***wait,mode*

where the parameters are the same as defined for the OPEN macro.

Example of CCM READ

Read a block of data 45 words long from line 13 of logical unit 215. Set wait and mode to 0. The actual data block is defined by an LCB at address LCBCRT.

```
DCMLUN   EQU      215
CRTLINE  EQU      13
RDCRT    READ     LCBCRT,DCMLUN
         .
         .
LCBCRT   LCB      45,BUFADR,CRTLINE
```

6.2.5 WRITE Macro

The WRITE macro causes the CCM to output a block of data of a specified length.

label **WRITE** **lcb,lun,***wait,mode*

where the parameters are the same as defined for the OPEN macro, except that mode = 1 when writing an ITB in bisyn mode (refer to section 8.4.5.5 for more information on ITB).

Example of CCM WRITE

Write a block of data 45 words long on line 15 of logical unit 27 from BUF 2. Select immediate return.

```
SLCLUN   EQU      27
NOWAIT   EQU      1
SLLINE   EQU      15
         .
         .
WRITE3   WRITE   SLCLCB,SLCLUN,NOWAIT
         .
         .
         .
SLCLCB   LCB      45,BUF2,SLLINE
```

6.2.6 FUNC Macro

The FUNC macro performs functions specific to the driver and hardware that cannot be handled with other macros.

label **FUNC** **lcb,lun,***wait*

where

lcb is the address of the line control block

lun is the number of the logical unit used to reference the CCM

wait 1 for immediate return or a zero (default) for suspension of the caller until request function is complete

Function	Function Code	Comments
Get latest status	0	Immediate return (see note 1).
Clear read request	1	Dequeues and sets error status on active request for the line (see note 1).
Clear write request	2	See note 1.
Sense event	3	See LCB description for specific events.
Reverse channel transmit ON	4	
Reverse channel transmit OFF	5	
Transmit break	6	Transmits break characters.
Load control characters	7	Loads (extension word) into LSD.
Answer line	8	Turn Data-Terminal-Ready ON.
Hang up line	9	Turn Data-Terminal-Ready OFF.
Load sync characters	10	Loads (extension word) into transmit (byte 0) and receive (byte 1) sync bytes in LSD and loads the registers in the synchronous line adapter.
Set E/P flag in line service descriptor table (LSD)	11	Asynchronous line adapter (LAD), enable hardware echo on receive. Synchronous LAD, select 8-bit (no parity) data byte format. Bisynchronous mode, accept ITB as regular characters, and input to memory.

(continued)

Function	Function Code	Comments
Reset E/P flag LSD	12	Asynchronous LAD, disable hardware echo on receive. Synchronous LAD, select 7-bit (with parity) data byte format. Bisynchronous mode, ITB is not input to memory.
Set DS/S flag in LSD	13	Asynchronous LAD (with modems that support dual speed feature), select higher speed operation. Synchronous LAD, do not store received sync bytes in memory. Bisynchronous mode, enables the sync-line feature on some Bell modems. It also causes a one millisecond pulse to be output to the modem.
Reset DS/S flag in LSD	14	Asynchronous LAD (see above), select lower speed operation. Synchronous LAD, store received sync bytes in memory. Bisynchronous mode, disables function code 13.
Select control character recognition	15	Terminate READ operation if either of the two control characters are recognized in data stream or if byte count = 0.
Ignore control character recognition	16	Terminate READ operation if byte count = 0 only.
Resync for each READ (full-duplex, synchronous LAD)	17	Synchronous LAD only. Causes resync to occur for each READ (bit in LSD).
Do not resync for each READ (full-duplex, synchronous LAD)	18	Negates effect of function code 17.
Terminate I/O request (receive) if line error detected	19	Causes termination of READ request immediately when line errors (break, parity error or data overflow) are detected.
Terminate I/O request (receive) only if byte count = 0 or control characters are received.	20	Error status is reported only after request completion. Negates FUNC 19.
Kill I/O	21	All READ, WRITE, and FUNC requests queued against the line are terminated with I/O error code 1 (device down) extended status word bit 7 set (I/O clear occurred) and the physical line is marked down (see note 1).
Set ASCII mode	22	Forces bit 7 = 1 of each byte input for compatibility with software.
Clear ASCII mode	23	Bit 7 takes on value determined by line adapter.

(continued)

Function	Function Code	Comments
Initialize line	24	Performs all initialization required by hardware and software.
Set in buffer chain mode	25	Enable the system to receive input in the buffer chain mode (see note 2).
Reset buffer chain mode	26	Resets a system from buffer chain mode back to "normal" mode.
Set "no block check"	27	Do not check the BCC after receiving an ITB control character. On output, ITB is a regular character. (No BCC; see Note 3).
Reset "Block check"	28	Check BCC after receiving an ITB. (See Note 3.)
Set ASCII/not	29	Set in ASCII/not transparent mode (see note 3).
Set ASCII with transparent	30	Set in ASCII/with transparent capability mode (see note 3).
Set in EBCDIC	31	Set EBCDIC mode, both for regular and transparent capability (default mode (see note 3).
BSC receive poll mode	32	Sets system in BSC receive poll mode (see note 4).
Reset BSC receive poll mode	33	Resets system from BSC receive poll mode (see note 4).

NOTE:

1. Immediate functions, all others queued.

2. If not executed from a foreground task, results in an error.

3. Only used with a BSC line adapter. An indication is given when these functions are used on any adapter other than BSC. On output, the WRITE macro must use mode-1 (see section 8.4.5.5).

4. Function code 32 can be used only when V$POLL is 1 (see section 13.2.2). If V$POLL is not 1, a completion error is given and the function is not performed. The system remains in the normal mode.

Example

Turn on reverse channel on line 14 of logical unit 45.

```
FUNLUN   EQU      45
         .
         .
         .
REVFUN   FUNC     RCLCB,FUNLUN
         .
         .
         .
 RCLCB   LCB      0,0,4,14
```

Note: Refer to the LCB example (defining an LCB for use with a FUNC request) in section 6.2.1.

6.2.7 STAT Macro

The macro causes the status of an I/O request to be examined and control to be transferred to a user defined routine for the processing of errors.

Status Macro:

label	**STAT**	**req,err,aaa,bbb,busy**

Where:

req	is the address of the I/O macro.
err	is the address of the I/O error routine.
aaa	is the address of the routine to process ring detected, or reverse channel ON conditions.
bbb	is the address of the routine to process break conditions.
busy	is the address of the I/O-not-complete routine.

Example of a STAT Macro

Check STATUS on the request macro READTY. If the request is busy, jump to the routine DELAY. If an error has occurred, jump to the routine ERR. If ring detected, or reverse channel on, jump to RING. If break, jump to BREAK.

```
STATL1 STAT READTY,ERR,RING,BREAK,DELAY
```

SECTION 7
BUFFER CHAINING

7.1 INTRODUCTION

Buffer chaining is a method of dynamically assigning buffer areas for incoming data. It eliminates the need for allocating large buffer areas. When incoming data fills one buffer the input is switched to the next buffer in the chain. This allows the application program to begin processing the data in the first buffer while the next buffer in the chain is receiving data. When the data in the first buffer has been processed the buffer can be reassigned to the chain.

With buffer chaining only one READ command is required for each segment of input data without using a large portion of memory.

The interface between the application program and the CCM is accomplished mainly through the following:

a. Chain Header (CHR)

b. Interface Block Header (IBH)

7.1.1 Queuing Procedure

Buffer chaining employs a double pointer queue header. The two pointers are the front pointer (F) and the rear pointer (R).

The initial contents of a double pointer queue header is:

F	0
R	Address of F

Two routines are used to add and remove the addresses from the double pointer queue. The routines are called PUTQ and GETQ. The routines can be coded as macros or subroutines.

7.1.2 PUTQ

The PUTQ macro adds (or queues) a buffer whose address is in the X register to a queue whose header address is in the B register.

The front and rear pointers are updated accordingly.

Calling sequence (as a macro):

```
LDXI    (buffer address)
LDBI    (queue header address)
PUTQ
```

Exit conditions:

A register = zero
X register = no change
B register = no change

Macro Code:

```
PUTQ    MAC
        STXE*   1,B
        STX     1,B
        TZA
        STA     0,X
        EMAC
```

Subroutine Code:

```
PUTQ    ENTR
        STXE*   1,B
        STX     1,B
        TZA
        STA     0,X
        JMP*    PUTQ
```

7.1.3 GETQ

The GETQ macro removes (dequeues) the first item from a double header queue whose address is in the B register.

Calling sequence (as a macro):

```
LDBI    (queue header address)
GETQ
```

Exit conditions:

A register = zero
B register = no change
X register = zero if queue was empty; or
address of item dequeued

Macro code:

```
GETQ    MAC
        LDX     0,B
        JXZ     *+7
        LDA     0,X
        STA     0,B
        JANZ    *+3
        STB     1,B
        EMAC
```

Subroutine Code:

```
GETQ    ENTR
        LDX     0,B
        JXZ     *+7
        LDA     0,X
        STA     0,B
        JANZ    *+3
        STB     1,B
        JMP*    GETQ
```

7.2 CHAIN HEADER

The chain header (CHR) contains the pointers of the active and complete chains. Each set of pointers is made up of two addresses, the front pointer and the rear pointer.

The active chain contains the pointers to the interface buffer headers (section 7.3) that contain the addresses of the chained buffers that are empty or in the process of being filled with input data. The complete chain contains the pointers to the interface buffer headers that contain the addresses of the chained buffers that are full and waiting to be processed by the application program.

The active chain front pointer contains the beginning address of the first interface buffer header in the active chain. The active chain's rear pointer contains the beginning address of the last interface buffer header in the active chain. The complete chain contains the pointers to the first and last interface buffer headers in the complete chain.

Note: Because both VTAM and the application program utilize the chain header, interrupts must be disabled before any buffers are added or removed from the chain header. The interrupts should be enabled immediately after the buffers have been added or removed.

The format of the chain header is described below:

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0	Active chain front pointer
1	Active chain rear pointer
2	Complete chain front pointer
3	Complete chain rear pointer
4	Address of request block

The chain header words should be initially set to the following values:

Word 0 = Zero

Word 1 = Word 0 address

Word 2 = Zero

Word 3 = Word 2 address

Word 4 = Zero

The initial values may be placed in the chain header by a user macro or by a direct data statement.

Examples:

a. user macro

```
CHR  MAC
     DATA     0,*-1,0,*-1,0
     EMAC
```

b. direct data statment

```
CHR  DATA     0,*-1,0,*-1,0
```

7.3 INTERFACE BLOCK HEADER

Each buffer presented to the CCM by the application program must be proceded by an interface block header (IBH).

The IBH is five words or more in length and defines the buffer area. It also contains a pointer to the next buffer in the chain.

The format of the IBH is described below:

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
0	Forward Pointer	IBH
1	Buffer Length	
2	Buffer Starting Address	
3	R/L Flag — Last Address	
4	Status	
	Variable User Area (optional)	
	Data Block Area	

Explanation of IBH fields:

Forward Pointer - Contains the address of the next IBH in the chain. When it is the last IBH in the chain, word zero contains all zeros.

Buffer Length - Contains the defined length (in bytes) of the data block attached to this IBH. (Buffers may vary in length, and may be tailored separately for each use.)

Buffer Starting Address - Contains starting address of the data area associated with this IBH.

R/L Flag - Signifies if the next free byte in the buffer is in the left or right half of the word.

0 = Right side (bits 0-7)
1 = Left side (bits 8-15)

Last Address - Contains the address of the next available word in the buffer. It is used with the R/L flag to determine where the next byte goes.

Status - Contains the status word of the IBH (used only by CCM).

Data Block Area - This is the buffer area that the data is read into. It may or may not immediately follow the IBH, but, if the buffer area immediately follows the IBH, it is easier to find any programming errors. The data areas (buffers) associated with different IBHs do not have to be the same size.

The IBH words should be initially set to the following values:

Word 0 = Any value (see note)

Word 1 = Length, in bytes, of the data area

Word 2 = Data area starting address

Word 3 = Zero

Word 4 = Zero

Word 5 = Zero

Note: Word 0 is filled by the PUTQ routine. Words 3, 4, and 5 are filled by the CCM.

A minimum of two interface block headers must be queued to the active chain of the chain header, at all times, to prevent the loss of incoming data.

Example of PUTQ Routine

Chain two previously defined IBHs (IBH1 and IBH2) for BUF1 and BUF2 to the active chain header at CHRO.

```
LDXI    IBH1
LDBI    CHR0
PUTQ
LDXI    IBH2
PUTQ
```

After execution, the chain header forword pointer contains IBH1 and the rear pointer contains IBH2. Word 0 of the IBH1 contains IBH2 and word 0 of IBH2 contains 0. Figure 7-1 shows the content of the active chain header and IBHs after the PUTQ routine is executed.

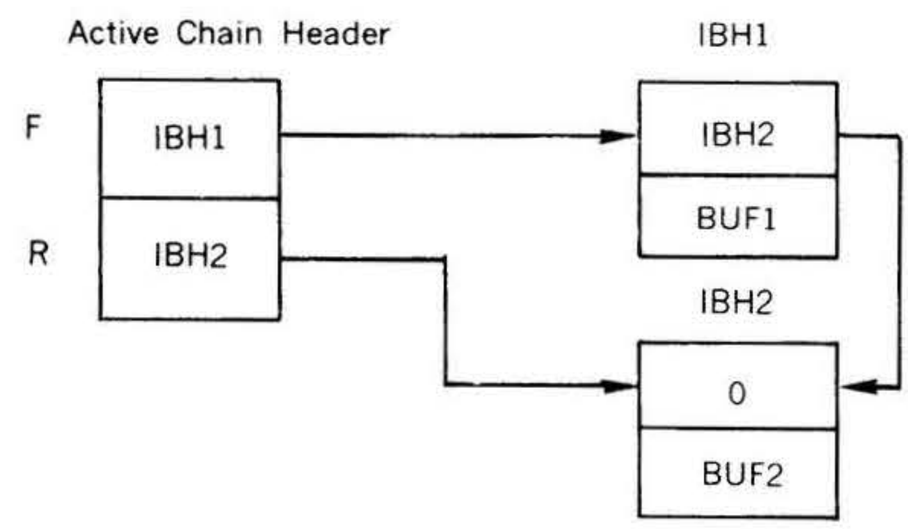


Figure 7-1. Contents of CHR and IBHs after PUTQ

Figure 7-2 shows the contents of CHR and IBH's before and after the first data block has been filled with a buffer chain mode READ.

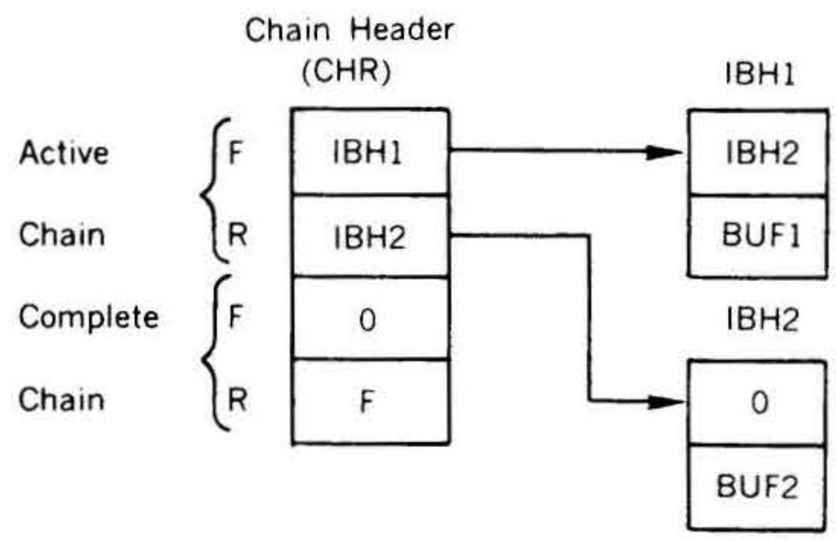


Figure 7-2. Contents of CHR and IBHs During READ

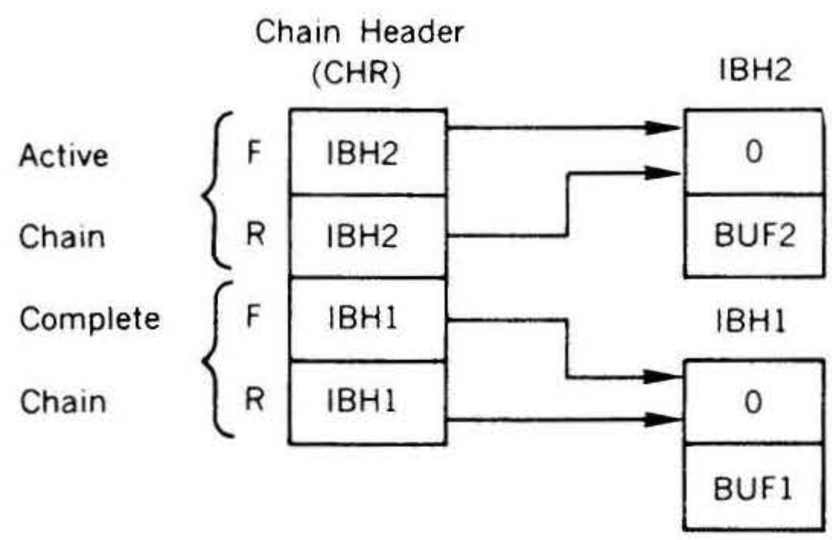


Figure 7-3. Contents of CHR and IBHs Before and After GETQ

Example of GETQ Routine

Remove the full buffer (BUF1) from the complete chain header CHR0.

```
LDBI      CHR0+2
GETQ
```

The data in the buffer area is now ready to be processed by the application program. The X register contains the address of the IBH for BUF1. Figure 7-3 shows the contents of the complete chain header and IBH before and after the GETQ routine is executed. Figure 7-4 shows the relationship of the various fields in the CHR and IBHs.

7.4 SET AND RESET FUNCTIONS

Function 25 is used to set the system in buffer chaining mode. Function 26 is used to reset the system from buffer chaining mode to normal mode.

When a FUNC 25 is issued, the set routine first validates that the calling application is a foreground task. If the calling application is a background task an error indication is generated and the request is terminated.

The error indication is set in word 2, bits 5-14 of the macro as follows:

CC = 5 (bits 5-7)
e = 1 (bit 8)
Status = 4 (bits 9-14)

(See section 3 for macro expansion description.)

If the calling application is a foreground task, the chain mode flag in the LSD is set to 1.

Note: Any READ request that is issued prior to a FUNC 25 being executed, is assumed to be in the normal mode.

When a FUNC 26 is executed, the system is restored to the normal mode and the chain mode flag in the LSD is set to zero.

Note: A FUNC 26 will not create an error if issued in the normal mode.

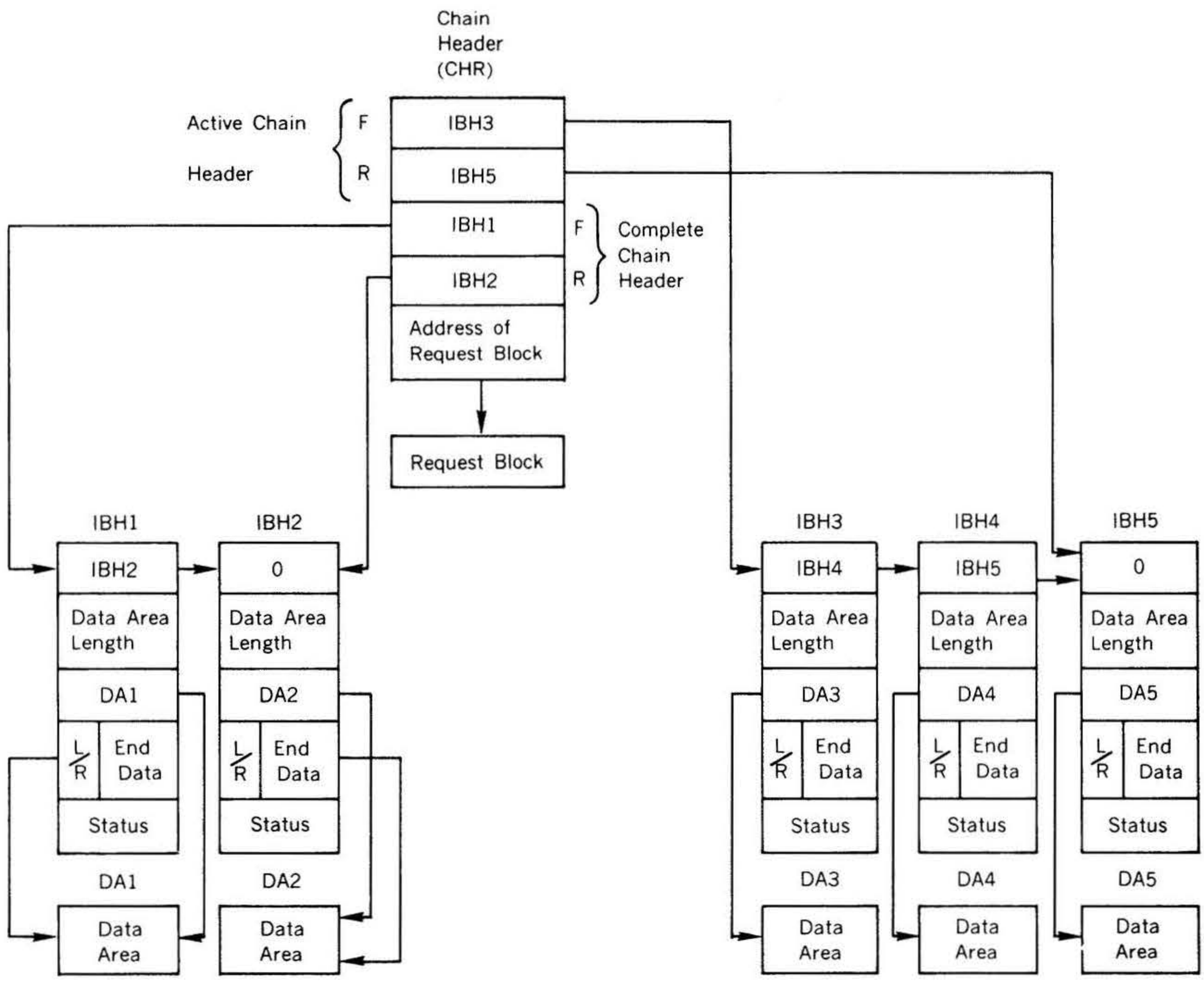


Figure 7-4. Relationship of CHR and IBHs

7.5 PROCEDURE FOR CODING A BUFFER CHAIN

In summary, the following steps should be taken in coding a buffer chaining routine:

a. Set the system in buffer chain mode (FUNC 25).

b. Chain some IBHs (minimum of two) to the active chain headers (PUTQ).

c. Issue a READ command with an immediate return followed by a DELAY with a type parameter of two. The LCB for this READ command must contain the address of the first IBH on the active chain as the buffer address. The record length field must contain the buffer byte count (with the byte count field set for the size of the first buffer). The LCB extension word must contain the address of the CHR.

Note: The delay is to notify the application program when the first buffer is full (if desired). This will allow the program to process the first part of a message without waiting for the EOT.

d. Clear the event word and bit 6 in the status word of the TIDB.

e. Remove all filled buffers from the complete chain to be processed by the application program (GETQ).

f. Supply enough buffers to the active chain whenever possible.
Note: The input operation is considered complete in one of two cases:
1. Control character is detected.
2. Active chain is empty.

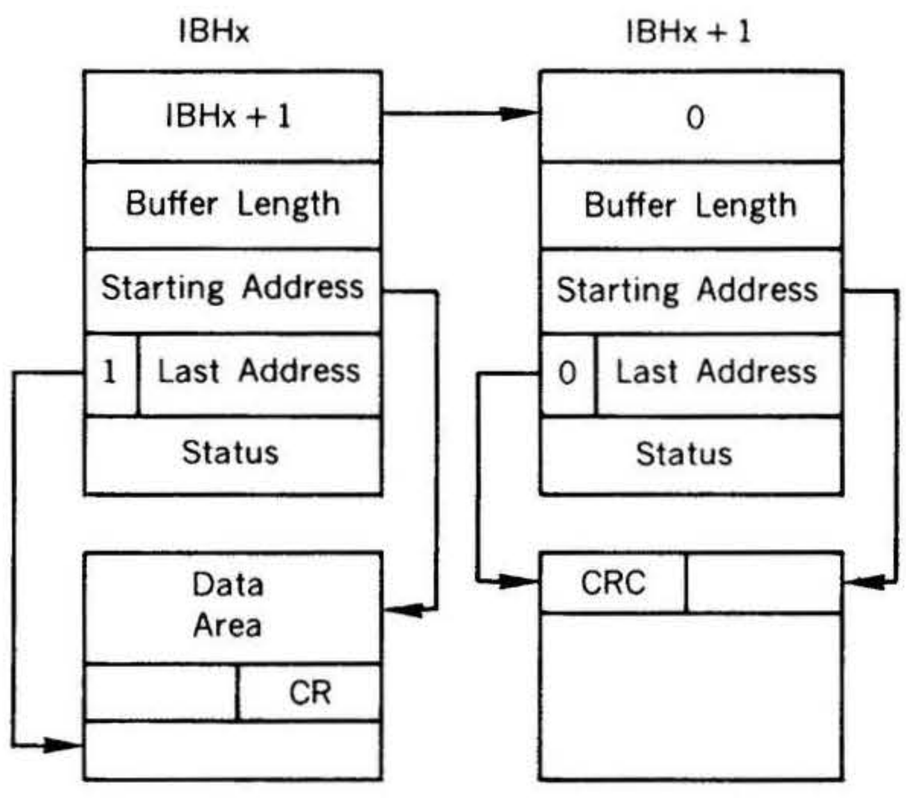
g. Test for input complete by examining the status word in the READ macro. (Use STAT macro.)

h. Remove the last block of data from the complete chain (GETQ).

Note: If the CRC-STOP attribute was defined (other than zero) in the line statement (section 2), the data and the EOM character will be in the data area. The additional characters of the CRC will be in the next higher referenced data area.

Example:

The EOM character was a CR and 1 additional character was specified to be read. The result is as follows:



SECTION 8
BINARY SYNCHRONOUS COMMUNICATION

8.1 INTRODUCTION

The Binary Synchronous Communications (BSC) procedure provides a set of rules for synchronous transmission of binary coded data. BSC expands the transmission capabilities of VTAM through its ability to accomodate a variety of transmission codes. BSC also has a transparent mode that allows transmission of control characters and various forms of raw data within the normal message format without any associated control or graphic significance. BSC is capable of accommodating a broad range of medium- and high-speed equipment.

All data in BSC is transmitted as a serial stream of binary digits. Synchronous communications means that the receiving station on a communications channel operates in step with the transmitting station through the recognition of a specific bit pattern (sync pattern) at the beginning of each transmission.

8.2 DATA LINK

A data link consists of the communications lines, modems, and other communications equipment arranged for data, used in the transmission of information between two or more stations.

All transmissions are sent over the line as a sequence of binary-coded signals. Control of the data link is accomplished by the transmission and recognition of special line-control characters.

The data link can be designed to operate either point-to-point (two stations) or multipoint (two or more stations).

8.2.1 Point-To-Point Data Link

A point-to-point data link consists of a communications facility between only two stations. All transmissions over the data link must be between the two stations operating on the data link. The point-to-point link can be established over leased (nonswitched) communications lines or a switched network. On a leased line (permanent-type connection), the transmissions are always between the same two stations. On a switched network, the data link is disconnected after the two stations complete their transmissions. A new data link is created for each subsequent transmission by standard dialing procedures (manual or automatic). The new data link may be established with any other station in the network.

8.2.2 Multipoint Data Link

For multipoint operation, one station in the network is always designated as the control station. The remaining stations are designated as tributary stations. The control station manages all transmissions within the multipoint data link, which is normally established over leased (nonswitched) lines. This is called a centralized multipoint operation. The control station initiates all transmissions by selecting or polling a tributary station. Any transmission over the data link is between the designated control station and one of the tributary stations. The other stations in the network are in a passive monitoring mode.

8.3 TRANSMISSION CODES

The major function of BSC is to effect the orderly transfer of data from one location to another using communications facilities. This data is transferred as binary-coded characters comprising text information (message body) and optional heading information (message identification and destination). In addition, data-link control characters are required with each message to delimit various portions of the message and control its transmission.

BSC can accommodate two different code sets (EBCDIC and ASCII). Both code sets may also be used in the transparent mode.

When either of these code sets is used with transparent mode, the flexibility of the telecommunications system is further increased since all possible bit configurations are treated as "data only" within transparent text. For this mode of operation, all assignment restrictions are removed from the code set being used. Thus the parity bit is also available as a data bit when transmitting ASCII-coded data in transparent mode. This additional BSC capability means that within the standard message format, any type of coded information can be handled using transparent mode.

Three functions are available to condition the system to one of the following modes:

FUNC 29

Conditions the system to operate in the "ASCII/not transparent" (ANT) mode. In this mode the ASCII character set is used. The switching to transparent mode is not possible.

FUNC 30

Conditions the system to operate in the "ASCII/with transparent capability" (AWT) mode. In this mode the ASCII character set for both message and control is used. When in AWT mode, the sequence DLE STX when detected will switch the system into the transparent mode, while either DLE ETX or DLE ETB when detected, will switch the system back to normal mode.

FUNC 31

Conditions the system to operate in the EBCDIC mode. This mode is similar to the AWT mode in which switching back and forth from normal to transparent is possible. The character set used is EBCDIC. The EBCDIC mode is the default one; i.e., when only this mode is being used, no FUNC is needed.

8.4 OPERATION OF THE DATA LINK

In point-to-point operation a contention situation exists, whereby both stations can attempt to use the communications line simultaneously. To minimize this possibility, a station bids for the line using the ENQ (enquiry) control character. The SYN SYN ENQ sequence (SYN SYN represents the synchronous idle characters) provides a concise signal for requesting control of the line, and thus leaves a maximum amount of time for line monitoring. If simultaneous bidding occurs, one station must persist in its bidding attempt to break the contention condition. Once a station gains control of the line, message transmission can start.

8.4.1 Polling and Selection

In a multipoint environment, the control station either polls or selects the tributary stations. Polling is an "invitation to send" transmitted from the control station to a specific tributary station. Selection is a "request to receive" notification from the control station to one of the tributary stations instructing it to receive the following message(s). These capabilities permit the control station to specify the transmitting station and to control the direction of transmission. Each station in the data link is assigned a unique station address, which is used to acquire the station's attention during either polling or selection. Each station address can consist of from one to seven characters, depending on the specific station requirements. The first character addresses the station itself, while additional characters indicate the desired component of the station. Depending on the particular station, the station address may consist of the first two characters, where the first character is repeated for increased reliability. Once the station's attention is acquired and it responds affirmatively message transmission can start.

9022 only!

Two FUNC macros are available for use with the BSC receive poll mode. The VTAM system enters and exits the poll mode with FUNC 32 and FUNC 33, respectively. While in the poll mode, only one READ command is required to receive a poll message.

The general format for the BSC receive poll message is:

SYN	SYN	EOT	PAD	SYN	Poll message	ENQ	PAD

EOT and ENQ are both either in ASCII or EBCDIC. The PAD and all SYN fields are dropped and the remaining fields are stored in the input buffer. The format of the input buffer is:

EOT	Poll message	ENQ

Any BSC control character may be used instead of EOT as long as the first control character (EOT in the above example) is not the same as the last one (ENQ).

FUNC 7 loads the control character configuration into the DCM. The FUNC 7 word format consists of control characters in the left byte and zeros in the right byte.

After successfully receiving the poll message, the poll mode is turned off using FUNC 33.

Examples:

Using EBCDIC, the EOT character set is 0 011 011 1. If LU0 is the logical unit for the DCM and LL0 is the BSC line number, the following statement defines the control character as EOT:

```
        FUNC        LCB7,LU0

LCB7    LCB         0,0,LL0,7,0,033400
```

The following statement switches the VTAM system to the BSC receive poll mode:

```
        FUNC        LCB 32, LU0

LCB32   LCB         0,0,LL0,32
```

The following statement switches the VTAM system back from the BSC receive poll mode:

```
        FUNC        LCB33,LU0

LCB33   LCB         0,0,LL0,33
```

The following sequence of events occurs during the BSC receive poll mode:

a. Load EOT,0:

```
FUNC        LCB7,LU0        (FUNC 7)
```

b. Switch VTAM system to the poll mode:

```
FUNC        LCB32,LU00      (FUNC 32)
```

c. Determine if FUNC 32 is executed:

```
READ poll message:
READ LCBRD,LU0,1            (read)
```

d. Switch VTAM system back from the poll mode:

```
FUNC        LCB33,LU0       (FUNC 33)
```

e. Resume normal operation

The VTAM system is interrupted by the DCM for every known control character (such as EOT,0). Thus when speed is essential, FUNC 7 can be used again (in step e above) instead of EOT,0 to load another byte configuration.

8.4.2 Message Blocks

The message consists of one or more blocks of text data. The text is transmitted in blocks to provide more accurate and efficient error control. The text data is the body of the message and is identified by a start-of-text (STX) character immediately preceding each block of text. In addition, each block of text except the last is immediately followed by an end-of-transmission-block (ETB) character or an intermediate block (ITB) character. The last block of text in a message is immediately followed by an end-of-text (ETX) character. Figure 8-1 shows an example of a regular message format.

The text of the message can be preceded by a heading that contains auxiliary information (e.g., station control, priority, etc.) pertaining to the following text data. The heading is identified by a start-of-heading (SOH) character immediately preceding it.

For greater reliability, a unique character should always follow SOH to identify the heading function. The reason for this is to preclude the possibility of heading data being interpreted as text data, or vice versa, due to transmission errors. This unique character should not be used following STX. The percent (%) character should not be used for this purpose, as SOH is presently used to identify request-for-test or station-dependent control messages.

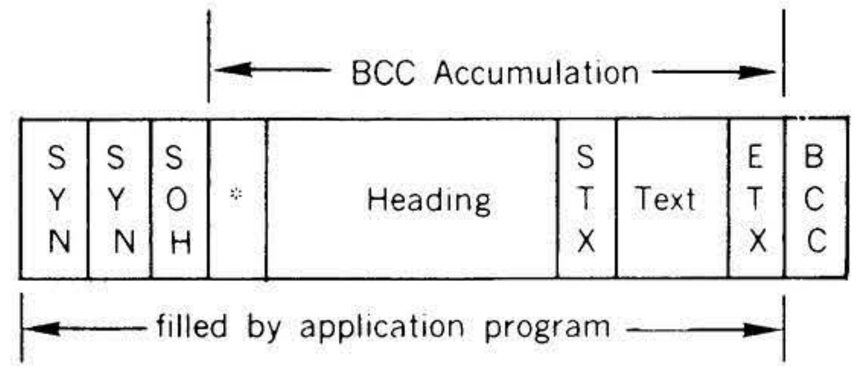


Figure 8-1. Regular Message Format

As each message block is completed, it is checked for transmission accuracy at the receiver before the transmission continues.

8.4.3 Error Checking

Each block of data transmitted is error-checked at the receiving station in one of two ways, depending on the code set being used (figure 8-2). These checking methods are longitudinal-redundancy checking (LRC) or cyclic-redundancy checking (CRC).

LRC is a longitudinal-redundancy check on the total data bits by message block. It is a basic form of CRC. An LRC character is accumulated at both the sending and receiving terminals during the transmission of a block. This accumulation is called the block-check character (BCC), and it is transmitted immediately following an ETB, ETX, or ITB character. The transmitted BCC is compared with the accumulated BCC character at the receiving station for an equal condition. An equal comparison indicates a good transmission of the previous block.

The LRC accumulation is reset by the first STX or SOH character received after a line turnaround. All characters received thereafter, including control characters, until the next line turnaround, are included in the accumulation. Only SYN characters are not accumulated. Following an ITB BCC, the accumulation resets and starts again with the next received STX or SOH character.

A cyclic-redundancy check is a division performed by both the transmitting and receiving stations using the numeric binary value of the message as a dividend, which is divided by a constant. The quotient is discarded, and the remainder serves as the check character, which is then transmitted as the block check character (BCC) immediately following a checkpoint character (ITB, ETB, or ETX). The receiving station compares the transmitted remainder to its own computed remainder, and finds no error if they are equal.

BCC is accumulated, sent, and checked on the receiving end by the BSC hardware. BCC errors are indicated by VTAM/CCM to the application program as parity errors.

8.4.4 EOT/NAK Pad Format Check

All BSC stations use the EOT/NAK pad format check to reduce the probability of a transmission line error converting an affirmative response (DLE sequence) into an EOT or NAK character. EOT and NAK must be followed by a trailing pad character of all "1" bits. Although all eight bits of the trailing pad character may be sent, the receiver should check only the first four bit positions. A station receiving an EOT or NAK within the text or heading of a transmission block (following STX or SOH) will treat the character as data and continue to receive or monitor the transmission (timeout, recognition of a turn-around character, etc.). The pad character is inserted by the BSC hardware.

Similar pad format checking on DLE sequences and ENQ may be done on an optional basis.

8.4.5 Data Link Control

Control of the data link is maintained through the use of control characters. Several variations in the designations and compositions of the data-link control characters and sequences exist between the two code sets. There variations are shown in table 8-1.

Table 8-1. Control Characters

Control Character	ASCII	EBCDIC
	76543210	
SYN	0001 0110	0011 0010
STX	0000 0010	0000 0010
DLE	0001 0000	0001 0000
ETX	0000 0011	0000 0011
ETB	0001 0111	0010 0110
SOH	0000 0001	0000 0001
ENQ	0000 0101	0010 1101
NAK	0001 0101	0011 1101
ITB	0001 1111	0001 1111
EOT	0000 0100	0011 0111
ACK 0	DLE 0	DLE '70
ACK 1	DLE 1	DLE/
WACK	DLE ;	DLE.
RVI	DLE <	DLE @
TTD	STX ENQ	STX ENQ

8.4.5.1 SYN - Synchronous Idle

This character is used to establish and maintain synchronization and as a time fill in the absence of any data or other control characters. Two contiguous SYNs at the start of each transmission (SYN SYN) are referred to as the character-phase sync pattern.

8.4.5.2 SOH - Start of Heading

This character precedes a block of heading characters. A heading consists of auxiliary information (such as routing and priority) necessary for the system to process the text portion of the message.

8.4.5.3 STX - Start of Text

This character precedes a block of text characters. Text is that portion of a message treated as an entity to be transmitted through to the ultimate destination without change. STX also terminates a heading.

8.4.5.4 ETB - End of Transmission Block

The ETB character indicates the end of a block-of-characters started with SOH or STX. The blocking structure is not necessarily related to the processing format. The block-check character is sent immediately following ETB. ETB requires a reply indicating the receiving station's status (ACK 0, ACK 1, NAK, or, optionally, WACK or RVI).

8.4.5.5 ITB - End of Intermediate Transmission Block

The ITB character is used to divide a message (heading or text) for error checking purposes without causing a reversal of transmission direction. The block-check character immediately follows ITB and resets the block-check count. After the first intermediate block successive intermediate

Transmission Code	Type of Checking		
	NO Transparency	Transparency Installed and Operating	Transparency Installed But Not Operating
EBCDIC	CRC-16	CRC-16	CRC-16
ASCII	LRC	CRC-16	CRC-16

Figure 8-2. Error Checking Capabilities

blocks need not be preceded by STX or SOH. (For transparent data, each successive intermediate block must begin with DLE STX and ITB must be the last character in the intermediate block.) If one intermediate block is a heading and the next intermediate block is text, STX must begin the text block.

Normal line turnaround occurs after the last intermediate block, which is terminated by ETB or ETX (DLE ETB or DLE ETX for transparency). When one of these ending characters is received, the receiving station responds to the entire transmission. If a block-check error is detected for any of the intermediate blocks, a negative reply is sent, which requires retransmission of all intermediate blocks.

All BSC stations must have the ability to receive ITB and its attendant BCC. The ability to transmit the ITB character is a station option. The ITB when sent, must be the last physical byte of the data block and the WRITE macro must be in mode 1.

8.4.5.6 ETX - End of Text

The ETX character terminates a block of characters started with STX or SOH and transmitted as an entity. The block-check character is sent immediately following ETX. ETX requires a reply indicating the receiving station's status.

8.4.5.7 EOT - End of Transmission

This character indicates the end of a message transmission, which may contain one or more blocks, including text and associated headings. It causes a reset of all stations on the line. EOT is also used as:

a. A response to a poll when the polled station has nothing to transmit.

b. An abort signal to indicate a system malfunction or operational situation that precludes continuation of the message transmission.

8.4.5.8 ENQ - Enquiry

The ENQ character is used to obtain a repeat transmission of the response to a message block if the original response was garbled or was not received when expected. ENQ is also used to bid for the line when using a point-to-point line connection. It also indicates the end of a poll or selection sequence.

8.4.5.9 ACK 0/ACK 1 - Affirmative Acknowledgment

These replies, in proper sequence, indicate that the previous block was accepted without error and the receiver is ready to accept the next block of the transmission. ACK 0 is the positive response to selection (multipoint) or line bid (point-to-point).

8.4.5.10 WACK - Wait-Before-Transmit Positive Acknowledgment

WACK allows a receiving station to indicate a "temporarily not ready to receive" condition to the transmitting station. It can be sent as a response to a text or heading block, selection sequence (multipoint), line bid (point-to-point with contention) or an ID (identification) line bid sequence (switched network). WACK is a positive acknowledgment to the received data block or to selection.

The normal transmitting station repsonse to WACK is ENQ, but EOT and DLE EOT are also valid responses. When ENQ is received, the receiving station will continue to respond with WACK until it is ready to continue. See the Continue Timeout discussion under Timeouts. An example of how WACK is used is shown in figure 8-3. The ability to receive WACK is mandatory for all BSC stations, but the capability to send WACK is optional.

8.4.5.11 NAK - Negative Acknowledgment

NAK indicates that the previous block was received in error and the receiver is ready to accept a retransmission of the erroneous block. It is also the "not ready" reply to station selection or line bid.

8.4.5.12 DLE - Data Link Escape

DLE is a control character used exclusively to provide supplementary line control characters, such as WACK, ACK 0, ACK 1, RVI, and transparent mode control characters. The sequences DLE STX, DLE ETX, DLE ITB, and DLE ETB initiate and terminate transparent text. In addition, other DLE control sequences (DLE ENQ, DLE DLE, DLE EOT) are used to provide active control characters within transparent text as required.

8.4.5.13 RVI - Reverse Interrupt

The RVI control sequence is a positive response used in place of the ACK 0 or ACK 1 positive acknowledgment. RVI is transmitted by a receiving station to request termination of the current transmission because of a high priority message which it must transmit to the sending station, or in case of a multipoint environment, the control station, acting as a receiver, now wishes to communicate with another station on the line. Successive RVIs cannot be transmitted, except in response to ENQ.

The sending station treats the RVI as a positive acknowledgment, and responds by transmitting all data that prevents it from becoming a receiving station. More than one block transmission may be required to empty the sending stations's buffers.

The character structure of the RVI control sequence is as follows:

EBCDIC	DLE@
ASCII	DLE<

The ability to receive RVI is mandatory for all BSC stations, but the ability to transmit RVI is optional. Figure 8-3 illustrates the use of RVI.

8.4.5.14 TTD - Temporary Text Delay

The TTD control sequence is sent by a sending station in message transfer state when it wishes to retain the line but is not ready to transmit. The TTD control sequence (STX ENQ) is normally sent after approximately two seconds if the sending station is not capable of transmitting the next text block or initial text block within that time. This two-second timeout avoids the nominal three-second receive timeout at the receiving station (figure 8-3).

The receiving station responds NAK to the TTD sequence, and waits for transmission to begin. If the sending station is still not ready to transmit, the TTD sequence can be repeated one or more times.

This delay in transmission can occur when the sending station's input device has not completely filled the buffer due to inherent machine timings. TTD is also transmitted by a sending station in message transfer mode to indicate to the receiver that it is aborting the current transmission (figure 8-3). After receiving NAK to this TTD sequence, the sending station sends EOT, resetting the stations to control mode (forward abort).

8.4.5.15 DLE EOT - Disconnect Sequence for a Switched Line

Transmission of DLE EOT on a switched line indicates to the receiver that the transmitter is going "on-hook." Either the calling or the called station may transmit this disconnect sequence. DLE EOT is normally transmitted when all message exchanges are complete, and may optionally be transmitted at any time instead of EOT to cause a disconnect.

Alternating Affirmative Acknowledgments

The BSC procedures specify the alternate use of ACK 0 and ACK 1 as affirmative replies. The use of ACK 0 and ACK 1 provides a sequential checking control for a series of replies. Thus it is possible to maintain a running check to ensure that each reply corresponds to the immediately preceding message block. ACK 0 is always used as the affirmative reply to selection or line bid.

8.5 MESSAGE FORMATS

There are three procedures involved in a basic message format, they are as follows:

a. Initialization procedure

b. Message transfer procedure

c. Termination procedure

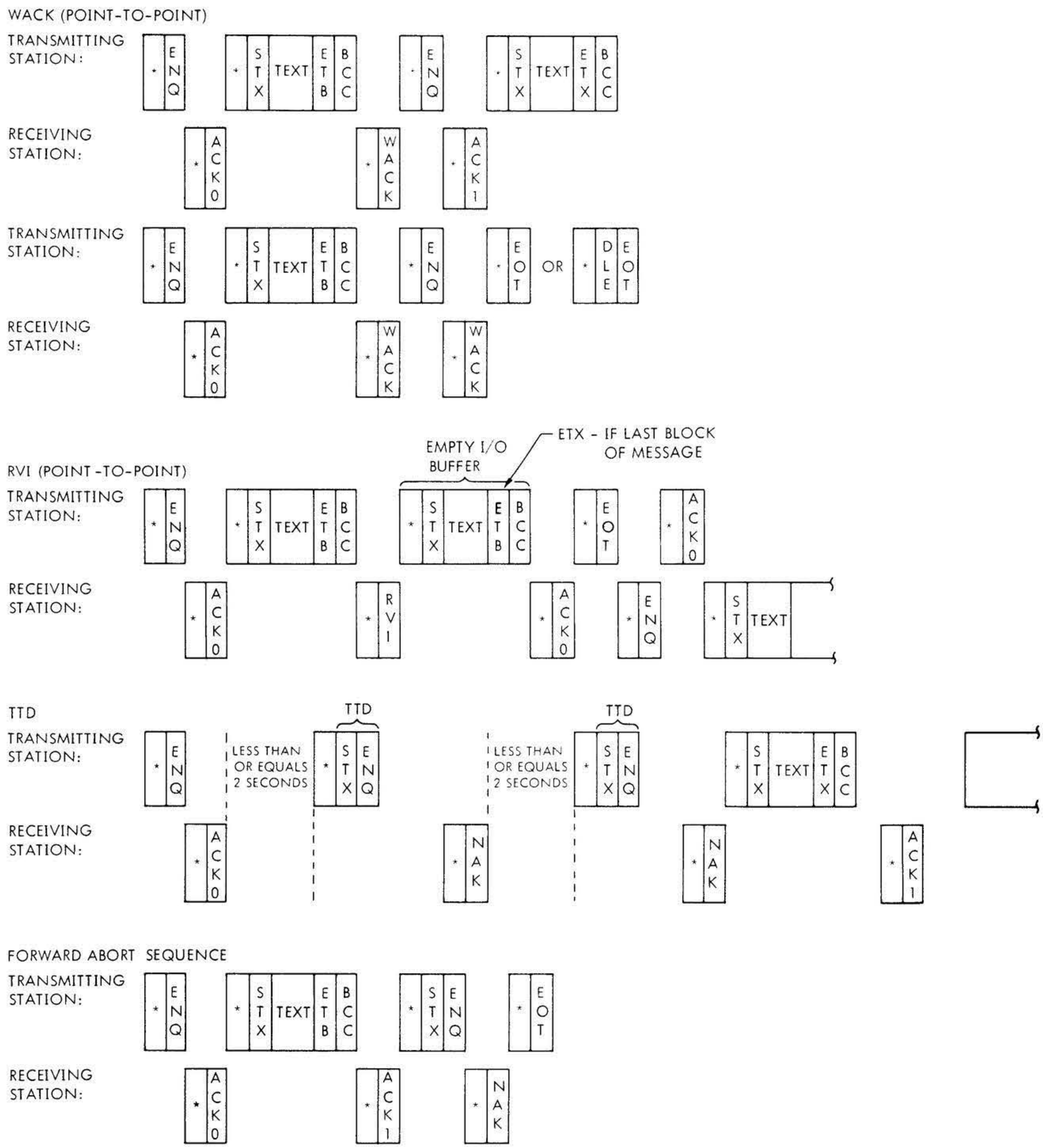
The binary synchronous communications discipline is based on a transmit-response philosophy of operation. That is, from the time that an initialization procedure commences on the communication line through to the termination procedure, there is a response to each turnaround character.

8.5.1 Initialization Procedure

The initialization procedure will consist of identification on a switched network, and of bidding on a point-to-point network.

8.5.1.1 Point-to-Point Operation (With Contention)

When transmission is started, an initialization sequence (ENQ character) is sent by the station attempting to acquire the line. The station receiving this character, and ready for input, replies with ACK 0. If the station is not ready for input it replies with NAK (Negative Acknowledgment). Simultaneous transmission problem is avoided by each station being assigned a priority. The high priority



VT12-416

Figure 8-3. Use of WACK, RVI, and TTD

station sends an ENQ to acquire the line and will continue to do so until an affirmative reply is received or until the retry limit is exhausted. The low priority station can only acquire the line if the high priority station has nothing to send.

8.5.1.2 Point-to-Point Operation (Without Contention)

In this mode of operation one station always starts the transmission whether it wants to output or request input. The master station sends the initialization sequence (ENQ). The slave station replies with the affirmative acknowledgment (ACK 0) if it is ready, or a negative acknowledgment (NAK) if it is not.

8.5.1.3 Dial Up Operation

Both stations start in circuit assurance mode. As soon as the dialed station goes "Off Hook" the dialing station sends one of the following messages:

-WRU - Who Are You

The sequence is ENQ.

-IAM - WRU -

The sequence is ID . . . ID . . . ENQ

The called station will reply with either:

-ID ACK - If ready

The sequence is ID....ID....ACKO

-NAK - If not ready

The sequence is ID....ID....NAK

The ID sequence is optional and consists of 1 to 7 characters of station identification. If the identification is incorrect either station can send a disconnect sequence.

8.5.2 Message Transfer Procedure

The message transfer procedure will begin with the first SOH or STX Character and ends with an EOT.

8.5.2.1 Transmitting Station

A message consists of one or more blocks of information. The start of text character (STX) precedes each block and the end of block character (ETB) followed immediately by the sumcheck character terminate that block.

The start of heading (SOH) followed by heading characters may precede the block of information. The End of text character (ETX) replaces the ETB for the last block of a message.

If transparent data is transmitted one DLE character directly precedes the STX characters (ETB or ETX must be the last character in the buffer). The transmitting station checks the response after each transmission block; further transmission sequence depends on the response from the receiving station:

a. A positive response (ACK 0/ACK 1) will result in sending of the next block of data.

b. A negative response (NAK) will result in the retransmission of the block.

c. No response (timeout) or a garbled response will result in a request for retransmission of the reply by sending an enquiry (ENQ).

8.5.2.2 Receiving Station

The receiving station replies to a transmission block with:

a. ACK 0 and ACK 1 - Alternately to indicate that the transmission was successful, and that it is ready for the next block.

b. NAK - To indicate that the transmission was erroneous and that it is ready for retransmission.

c. WACK - To indicate that the transmission was successful but that it is temporarily not ready to receive.

8.5.3 Termination Procedure

Message transmission is ended by the transmission of the end-of-transmission character (EOT). The station receiving the EOT can now bid for the line and become the transmitting station.

On a switched network, after completion of all message exchange, the mandatory disconnect (DLE EOT) can be sent by either station before disconnecting the line.

8.5.4 Transparent Mode

The system recognizes the sequence DLE STX as a request to switch to the transparent mode. The sequence ETX or ETB, as the last character in the buffer, switches the system back into a normal (ASCII or EBCDIC - as may be the case) mode. All data link control characters can be transmitted as transparent data without taking on control meaning.

Any data-link control characters transmitted during transparent mode must be preceded by a DLE to be recognized as a control function. Thus the following sequences are effective during transparent-mode operation:

Sequence	Use
DLE STX	Initiates the transparent mode for the following text.
DLE* ETB	Terminates a block of transparent text, returns the data link to normal mode, and calls for a reply.
DLE* ETX	Terminates the transparent text, returns the data link to normal mode, and calls for a reply.
DLE SYN	Used to maintain sync or as time-fill sequence for transparent mode.
DLE ENQ	Indicates "disregard this block of transparent data" and returns link to normal mode.
DLE DLE	Used to permit transmission of DLE as data when a bit pattern equivalent to DLE appears within the transparent data. One DLE is disregarded; the other is treated as data.
DLE* ITB	Terminates an intermediate block of transparent data, returns the data link to normal mode, and does not call for a reply. The block check character follows DLE ITB. Transparent intermediate blocks may have a particular fixed length for a given system. If the next intermediate block is transparent, it must start with DLE STX.

* The DLE part of the sequence is not placed in the buffer by the application program. When in transparent mode, ETX, ETB, or ITB are recognized by VTAM/CCM and sent as DLE ETX, DLE ETB, or DLE ITB only if they are the last character in the buffer.

The DLE STX following an intermediate transparent block may be preceded by SYN SYN, to permit any station out of sync to correctly synchronize with the transmission.

All replies, enquiries, and headers are transmitted in normal mode. Transparent data is received on a character-by-character basis; thus character phase is maintained in the usual manner.

An example of a block of transparent data is shown in figure 8-4.

The boundaries of transparent data are determined by the DLE STX and the ITB, ETB, or ETX sequences, which initiate and terminate the transparent mode. Thus, the length of a transparent message can vary with each transmission.

For checking the transmitted transparent data, CRC-16 is available. Refer to Error Checking for the available options. If the system has VRC in normal mode, this is suppressed within transparent-text blocks. This permits using the parity bit as an additional data-bit position for each character transmitted as transparent data.

Note: In transparent mode, the end control character (ETX, ETB or ITB) must be the last physical byte in the block of data. (The DLE and BCC will be inserted by the BSC hardware.)

8.5.5 Timeouts

Timeouts are used to prevent indefinite data-link tie-ups, due to false sequences or missed turnaround signals, by providing a fixed time within which any particular operation must occur. Due to the different requirments for the various operations, four specific timeout functions are provided: transmit, receive, ~~discount~~, and continue.
disconnect

8.5.5.1 Transmit Timeout

This is a nominal one-second timeout that establishes the rate at which sync idles are automatically inserted into

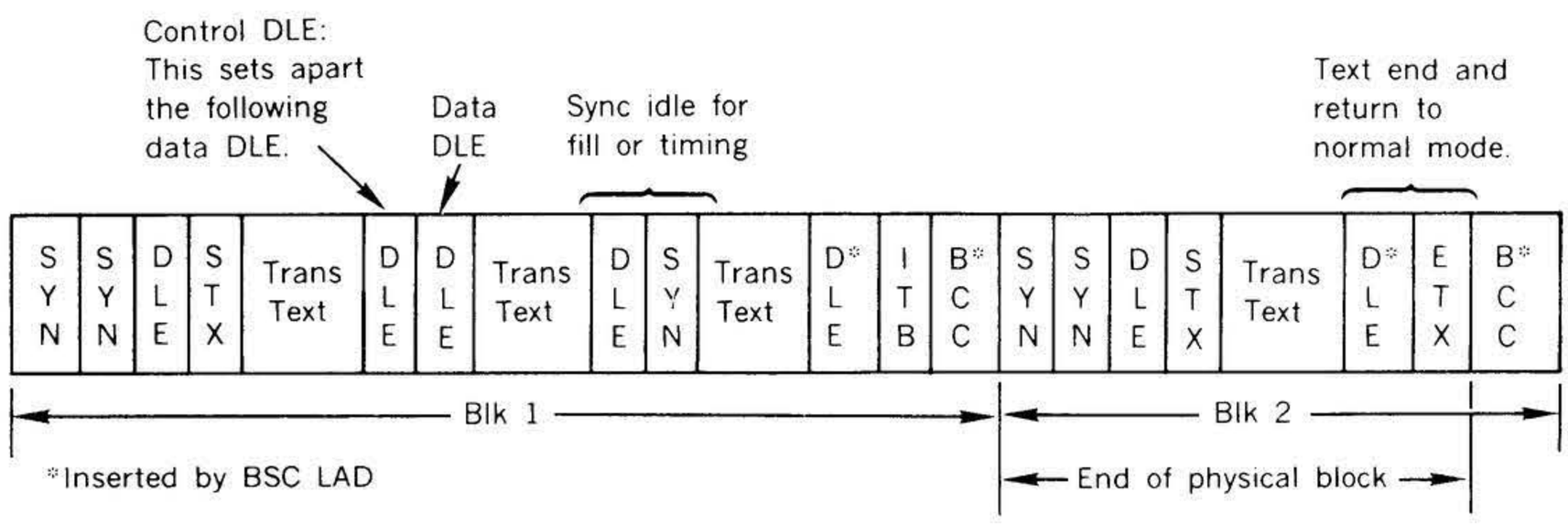


Figure 8-4. Transparent Data Block

transmitted heading and text data. In normal data, two consecutive sync-idle characters (SYN SYN) are inserted by the BSC hardware every second, while for transparent data, one transparent sync-idle sequence (DLE SYN) is inserted every second.

8.5.5.2 Receive Timeout

This is a nominal three-second timeout, and is used as follows:

a. Limits the waiting time tolerated for a transmitting station to receive a reply.

b. Permits any receiving or monitoring station to check the line for sync-idle signals. These sync idles indicate that the transmission is continuing; thus this timeout is reset and restarted each time a sync idle is detected.

c. Limits the time any tributary station in a multipoint network will remain in control mode while monitoring the line for its address code. This timeout runs whenever the station is in control mode. It is reset and restarted each time an end signal (EOT, ENQ, NAK, WACK, ACK) is recognized, as long as the station remains in control mode.
This timeout is done by hardware, and is monitored by VTAM/CCM. In case a three-second timeout occurrs, an error indication is returned via the request block. Both the parity error (bit 5) and overflow (bit 9) will be set in the Detailed Status.

8.5.5.3 Disconnect Timeout

This timeout is used optionally on switched network data links. It is a nominal 20-second timeout used to prevent a station holding a connection for prolonged periods of inactivity. After 20 seconds of inactivity, the station will disconnect from the switched network.

Note: The disconnect timeout function is not performed by VTAM/CCM, but may be implemented by the application program.

8.5.5.4 Continue Timeout

This is a nominal two-second timeout associated with the transmission of TTD and WACK. The continue timeout is used by stations where the speed of input devices (for transmitting stations) or output devices (for receiving stations) effect buffer availability and may cause transmission delays.

TTD is sent by the transmitting station up to two seconds after receiving acknowledgment of the previous block if the transmitting station is not capable of sending the next transmission block before that time.

Note: The continue timeout function is not performed by VTAM/CCM but may be implemented by the application program.

A receiving station must transmit WACK to indicate a "temporarily not ready to receive" condition if it is not able to receive within the two-second timeout. The purpose of the timeout interval is to permit the receiving station to send an appropriate affirmative reply immediately if it becomes appropriate within the interval.

8.5.6 Pad Characters

To ensure that the first and last characters of a transmission are properly transmitted by the data set, all BSC stations add a pad character before and after each transmission. The one-character pad (leading pad) preceding each initial synchronizing pattern ensures that the station will not start sending its synchronizing pattern before the other station is prepared to receive. The leading pad character is the sync character sent by the BSC hardware.

A pad character (trailing pad) is also added following each transmission (e.g., NAK, EOT, ENQ). Since ETB or ETX causes line turnaround, the pad character follows the BCC. The trailing pad character ensures that the last significant character (e.g., ETB BCC, ETX BCC, or NAK) is sent before the data set transmitter turns off. The trailing pad character consisting of all ones (hex 'FF') is sent by the BSC LAD.

8.6 TRANSMISSION SEQUENCE AND RECOVERY PROCEDURES

Table 8-2 shows examples of some of the transmission and recovery procedures.

Table 8-2. Transmission and Recovery Procedures

TRANSMISSION WITH CONTENTION

Terminal A (Priority 1)	Terminal B (Priority 2)
ENQ →	← ENQ
ENQ →	
	← ACK0
STX, MSG1, ETX →	
	← ACK1
EOT →	
	← ENQ
ACK0 →	
	← STX, MSGA, ETX
ACK1 →	
	← EOT

TRANSMISSION WITHOUT CONTENTION

Terminal ready to receive

Calling CPU	Called Terminal
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← ACK1
STX, MSG2, ETX →	
	← ACK0
EOT →	

Terminal Ready to Transmit

Calling CPU	Called Terminal
ENQ →	
	← ACK0
EOT →	
	← ENQ
ACK0 →	
	← STX, MSGA, ETB
ACK1 →	
	← STX, MSGB, ETB
ACK0 →	
	← EOT

POSITIVE RESPONSE

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← ACK1
STX, MSG2, ETX →	
	← ACK0
EOT →	

NEGATIVE RESPONSE

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← NAK
STX, MSG1, ETB →	
	← ACK1
STX, MSG2, ETX →	
	← ACK0
EOT →	

LINE FAILURE DURING RESPONSE

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← ACK0*
ENQ →	
	← ACK1
STX, MSG2, ETX →	
	← ACK0
EOT →	

* ACK1 Character changed to ACK0 due to line failure.

INVALID RESPONSE

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	XXX
ENQ →	
	XXX
ENQ →	
	XXX
ENQ →	
	XXX
EOT → (Disconnect)	

XXX = Invalid Response

Table 8-2. Transmission and Recovery Procedures *(continued)*

NO RESPONSE

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	TIME-OUT
ENQ →	
	TIME-OUT
ENQ →	
	TIME-OUT
ENQ →	
	TIME-OUT
EOT →	

TEMPORARY TEXT DELAY (TTD)

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← ACK1
TTD →	
	← NAK
TTD →	
	← NAK
STX, MSG2, ETX →	
	← ACK0
EOT →	

FORMAT ERROR CONDITION

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← ACK1
STX, MSG2, ETB →	
	← ACK1
ENQ →	
	← ACK1
ENQ →	
	← ACK1
ENQ →	
	← ACK1
EOT →	

REVERSE INTERRUPT (RVI)

Station A	Station B
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← RVI
*STX, MSG2, ETB →	
	← ACK0
EOT →	
	← ENQ
ACK0 →	
	← STX, MSGA, ETB
ACK1 →	
	← STX, MSGB, ETX
ACK0 →	

* I/O buffer is emptied before sending EOT.

OUT-OF-STEP CONDITION

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	TIME-OUT
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← ACK1
STX, MSG2, ETX →	
	← ACK0
EOT →	

WAIT BEFORE TRANSMIT POSITIVE RESPONSE (WACK)

Transmit	Receive
ENQ →	
	← ACK0
STX, MSG1, ETB →	
	← WACK*
ENQ →	
	← ACK1
STX MSG2, ETX →	
	← ACK0
EOT →	

* Message received correctly but no buffer available for second message.

Table 8-2. Transmission and Recovery Procedures *(continued)*

CIRCUIT ASSURANCE-GOOD-IDENTIFICATION

Station A	Station B
ID, ENQ →	
	← ID, ACK0
STX, MSG1, ETB →	
	← ACK1
STX, MSG2, ETX →	
	← ACK0
EOT →	

CIRCUIT ASSURANCE-STATION B IS NOT READY TO COMMUNICATE WITH STATION A

Station A	Station B
ID, ENQ →	
	← ID, NAK
ID, ENQ →	
	← ID, NAK
ID, ENQ* →	
	← DLE EOT Disconnect

* Number of retries is determined by the user.

SECTION 9

MANAGING BUFFERS

9.1 INTRODUCTION

VTAM provides three service routines to access temporary storage in central memory. The service routines are reentrant subroutines which are resident in central memory and have entry points in the VORTEX CL library.

The subroutines are VT$GTM, to acquire a block of temporary storage from a predefined memory pool, VT$PTM to return a block of temporary storage to a memory pool and VT$BMT to build a memory allocation table for a user.

9.2 MEMORY ALLOCATION ROUTINES AND THEIR FUNCTIONS

9.2.1 VT$BMT

A memory allocation table must be built for a memory pool to be accessed with VT$GTM and VT$PTM to allocate and deallocate its temporary storage blocks. VT$BMT creates the memory allocation table. VT$BMT is called by use of the VORTEX ALOC macro.

label **ALOC** **VT$BMT**

Before calling this subroutine, the user must load the A register with the size of the memory pool and the B register with the address of the memory pool. In addition the first locations of the memory pool must be set as follows:

Entry parameters

Memory Pool	Contents
+0	Smallest block size
+1	Number of blocks
+2	Next smallest block size
+3	Number of blocks
	.
	.
	.
2n-2	Largest block size (n)
2n-1	Number of blocks
+2n	0 zero
	Remainder of Memory pool

Exit Parameters

On return from the call, the memory pool will now have the memory allocation table in the first locations. The memory allocation table will have the following format:

Memory Pool Location	Contents
0	First block size
1	Head of queue
2	Second block size
3	Head of its queue
.	
.	
.	
2n-2	Nth block
2n-1	Head of its queue
2n	0

Error Indications on VT$BMT

On return, the status will be set in the A register. Zero indicates an error. The memory pool was not large enough to build the desired memory allocation table, or the block sizes were not in ascending order. When an error occurrs, the first word of the memory pool is set to zero.

Example:

Build a memory allocation table for a pool beginning at location BLKADR and extending 560 words. Specify 10 blocks of 20 words, 10 of 15 words and 20 of 10 words.

Prior to the VT$BMT call the first seven locations of the memory pool must contain the following:

Location	Value
+0	10
+1	20
+2	15
+3	10
+4	20
+5	10
+6	0

```
EXT   VT$BMT
  .
  .
  .
LDBI  BLKADR
LDAI  560
ALOC  VT$BMT
```

Upon return, the memory allocation table would appear as follows:

BLKADR

+0	10
+1	BLKADR +550
+2	15
+3	BLKADR +345
+4	20
+5	BLKADR+210
+6	0

In this example three memory locations (+7, 8, and 9) would be unused.

9.2.2 VT$GTM

The VT$GTM routine allows a user to acquire a block of temporary storage from a previously defined memory pool. If the memory allocation table for the pool does not have blocks of the specified size, the request is completed and an error is indicated by setting the A register to zero.

The VT$GTM routine is called by use of the VORTEX ALOC macro.

label **ALOC** **VT$GTM**

Before making the above call, the user must load the A register with the number of words in the block desired, and the B register with the address of the memory allocation table. The A register contains the address of the block upon return. The VT$GTM routine must not be called by a FORTRAN program since the contents of the register will not contain the desired parameters.

Error Indications

The status after a request to allocate memory is returned in the A register as follows:

A = 0	No blocks of the desired size are available
A ≠ 0	Address of the block (normal return)

The caller should be cautious in the use of this subroutine because invalid parameters could damage either the memory allocation table or other programs in the system.

EXAMPLE

Request a block of memory of 20 words from a pool maintained by memory allocation table MAT5.

```
EXT   VT$GTM
 .
 .
 .
LDAI  20
LDBI  MAT5
ALOC  VT$GTM
```

9.2.3 VT$PTM

The VT$PTM subroutine returns a specified-size block of temporary storage to a memory pool. If the memory allocation table for the pool does not contain blocks of the specified size, the next larger size in the memory allocation table will be used. This subroutine is called by use of the VORTEX ALOC macro:

label **ALOC** **VT$PTM**

Before making a VT$PTM call, the user must load the A register with the address of the memory allocation table for the pool, the B register with the address of the block being returned, and the first location of the block must contain the size of the block. Normal return is indicated by the A register equal to zero.

Error Indication

If the A register is not zero, then no block of the specified size was found to be deallocated.

Example:

Return a block of memory whose address is in location BLKADR which is 15 words long, to the pool maintained by a memory allocation table MAT5.

```
EXT   VT$PTM
 .
 .
 .
LDB   BLKADR
LDAI  15
STA   0,B
LDAI  MAT5
ALOC  VT$PTM
```

SECTION 10
CODING A TERMINAL CONTROLLER MODULE (TCM) FOR VTAM

10.1 INTRODUCTION

For each additional type of line service rule extending the VTAM system beyond the TTY TCM capabilities (described in section 5) a TCM must be written. For example a system which has Teletypes, synchronous CRT devices and a communications link to a large-scale processor involves three types of line disciplines, and so uses three TCM's.

In applications where little or no line discipline is required a user will not need to write a TCM because he may call the CCM directly. A TCM is useful where it can simplify a relatively complex line discipline.

A TCM is responsible for terminal unit control, error checking, code conversion and all other functions not handled by the CCM relating to control of the line and terminal equipment on the line. The main function of the TCM is to translate and break down the requests received from the application into a series of CCM requests which perform the particular line discipline. In effect a TCM handles the setting up of the CCM requests to perform a particular I/O operation whereas the CCM handles the actual I/O transfer.

In order to understand the function a Terminal Controller Module (TCM) performs in VTAM, one must trace the steps involved in building a VTAM system. The five main components of VTAM are: Network Definition Module, Network Control Module, Terminal and Line OPEN/CLOSE Processors, the Communication Controller Module (CCM), and the TCM. VTAM is designed to work with terminal-oriented tables called Terminal Controller Descriptors (TCD) and line-oriented tables called Line Service Descriptors (LSD). Since a TCM only works with terminal-oriented tables, only the TCD and its structure need to be described for coding a TCM.

10.2 TABLES USED BY TCM

During network definition, prototypes of TCD's are built by the NDL processor in a file called VT$DFT in the foreground library from terminal directives input to the NDL processor. These prototype TCD's are used by the Terminal Unit OPEN/CLOSE processor to build TCD's in central memory when a terminal is opened.

The Network Control Module, (NCM) through which a user can interrogate the status of the data communication network or alter it, is intimately related to the structure of the TCD, and as such, any changes to the TCD's structure should be kept to adding entries to it and keeping the current structure intact. As long as this restriction is followed, modifications to NCM may not be necessary.

The two major components that need to be considered when coding a TCM are the Terminal Unit OPEN/CLOSE Processor and the TCM Executive, (TCMEXEC). The function of the Terminal OPEN/CLOSE Processor is to build the TCD's and thread them to the proper VTAM tables. The TTY TCM is composed of a root segment, VT$OCT, and an overlay segment, TTYTCM, which is designed to build TCD's for the TTY TCM. To modify or extend the structure of the TCD, a new overlay segment must be written. The root segment, VT$OCT, keys on the TCM type PCTYP, from the prototype TCD in the VT$DFT file. All that is necessary to incorporate a new overlay segment is to write the overlay segment.

Ten TCM types are supported: one for the TTY TCM and nine for user-defined TCMs. The overlay names for the user-defined TCMs are TERM1M through TERM9M for TCM types 1 through 9, respectively.

In addition, return to the root segment should be made at VT$OCY or VT$OCZ depending on whether interrupts should be disabled or not. For example if interrupts are currently disabled in the overlay segment and interrupts are to be enabled, return should be made at VT$OCZ, otherwise return should be made at VT$OCY.

The following is a description of the current structure of the Terminal Controller Descriptor (TCD):

Field Label	Word	Bits	Description
TCTCD	0	0-15	Address of Next TCD in Queue
TCRQH	1	0-15	Head of Request Queue
TCCTA	2	0-15	Address Controller Table for TCM
TCCLN	3	0-7	LUN for the CCM
TCLLN	3	8-15	Logical Line Number
TCPCH	4	0-7	Prompt Character for Terminal
TCSWL	4	8-8	Switch/Non-Switched Flag
TCBSL	4	9-9	Sync/Asynchronous Flag
TCXMM	4	10-11	Transmission Mode
TCECH	4	12-12	Echo/No-Echo Flag

Field Label	Word	Bits	Description
TCCON	4	13-13	Physical Connection Flag
TCWBC	4	14-14	0 = Word Count, 1 = Byte Count for Write
TCRBC	4	15-15	0 = Word Count, 1 = Byte Count for Read
TCNTD	5	0-3	Number of Devices
TCNOD	5	4-7	Number of Devices Open
TCTYP	5	8-11	TCM Type (0 = TTY TCM)
TCCTP	5	12-15	Transmission Code Type (0 = ASCII)
TCRMD	6	0-2	Mode of Read Operation
TCWMD	6	3-5	Mode of Write Operation
TCRRS	6	6-8	Read Request Status
TCWRS	6	9-11	Write Request Status
TCLDF	6	12-12	Line Disconnect Flag
TCIBC	6	15-15	Input Buffer Chaining Mode
TCRCA	7	0-15	CCM Request Address for Read
TCSTO	8	0-15	Read Timeout Value
TCWCA	9	0-15	CCM Request Address for Write
TCDCC	10	0-15	Dynamic Character Count for Read
TCRBF	11	0-15	Dynamic Read Buffer Address
TCDTO	12	0-15	Dynamic Read Timeout Value
TCID1	13	0-15	First 2 Characters of TUID
TCID2	14	0-15	Second 2 Characters of TUID

When TCIBC is set (input buffer chaining mode), the following should be added to the TCD structure:

Field Label	Word	Bits	Description
TCACF	15	0-15	Active Chain Front CHR word 1
TCACR	16	0-15	Active Chain Rear CHR word 2
TCCCF	17	0-15	Complete Chain Front CHR word 3
TCCCR	18	0-15	Complete Chain Rear CHR word 4
Reserved	19	0-15	Reserved CHR word 5

After extensions to the structure of the TCD have been defined and the Terminal Unit OPEN/CLOSE Processor overlay segment designed to handle the changes to the TCD structure, the user must consider how to interface a new TCM with the VTAM system.

10.3 TCM FUNCTIONS

A TCM, in general, consists of two functional groups of programs – the VTAM TCM Executive (TCMEXEC) and a set of TCM request processing programs. The TCM Executive itself consists of an enqueuing module, VT$TCQ, and the TCM request initiation and completion module, TC$CEX, which is the main executive routine. I/O requests to a TCM are processed by IOC like I/O requests to standard VORTEX I/O drivers. When IOC processes an I/O request for a TCM, the request is queued against the TCM's controller table and the pseudo driver, VT$TCQ, is activated to queue the request to the proper TCD.

Figure 10-1 depicts the relationship of VTAM and TTY TCM modules.

When coding a TCM, one must consider how a TCM controller table (CTBL) should be structured. A TCM controller table is composed of two parts, the standard VORTEX controller table and the TCM Processor Table (TPT). The following is a description of the standard controller table part:

Entry	Word	Description
CTIDB	0	Controller Active Flag/TIDB Address
CTADNC	1	Controller Table End Plus One
CTOPM	2	Op code Mask, which is set to the sum of equate values for valid op codes for the TCM.
CTDST	3	Address of DST (= 0, set by IOC)
CTRQBK	4	Address of Request Block to be Processed. (= 0, set by IOC)
CTRTRY	5	Not used, set to 0.
CTDVAD	6	Controller Device Address
CTIOA	7	I/O Algorithm
CTSTAT	8	= 0, for TCM use
CTBICB	9	Not used, set to 0.
CTFCB	10	= 0, (set by IOC)

Entry	Word	Description
CTWDS	11	= 0, for TCM use.
CTFRCT	12	I/O Algorithm Frequency Count

The second part of the TCM CTBL is the TCM processor table, which should be changed according to the needs of the TCM. An example of a TCM processor table, for the TTY TCM, is the following:

Entry	Word	Description
TPRPA	13	Primary entry point to TTY TCM Read request processor program.
TPWPA	14	Primary entry point to TTY TCM Write request processor program.
TPFPA	15	Primary entry point to TTY TCM Function/WEOF processor program.

A possible extension to the TCM controller table would be to keep the standard part constant and to add additional entries to the TCM processor table for new TCM request processing programs.

10.4 TCM COMPONENTS

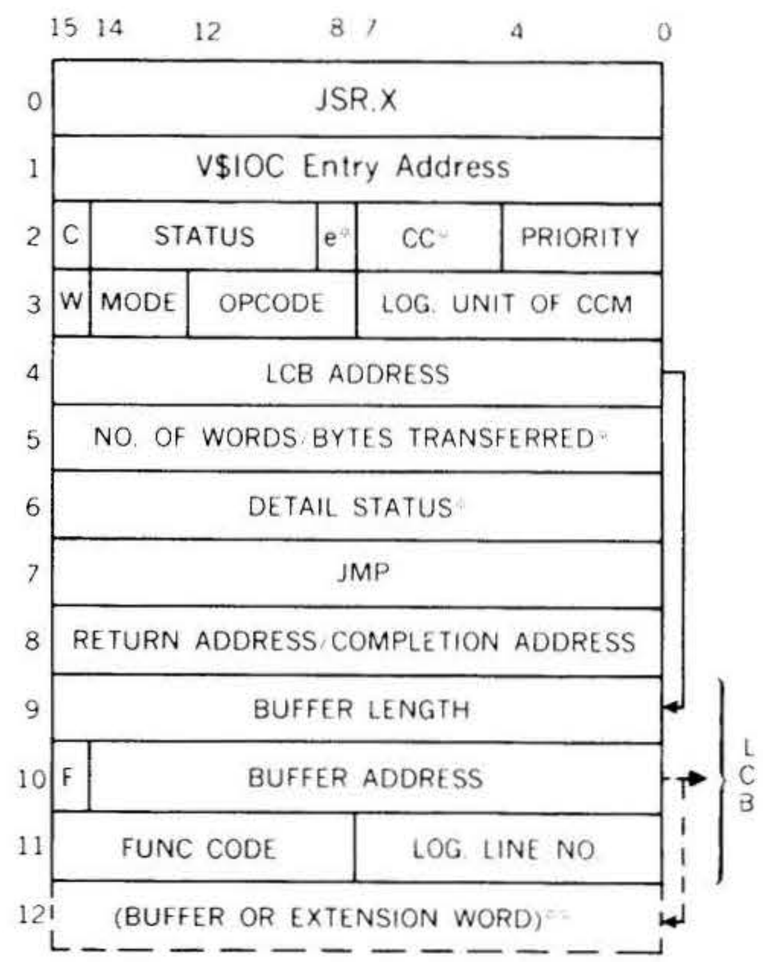
With an understanding of how the TCM controller table should be structured, the user can now consider how the different components of a TCM work together.

VT$TCQ, the enqueuing module, is responsible for queuing a TCM request on the proper TCD request queue from the TCM controller table. A TCM is referenced by a logical unit number that has been assigned to the TCM. A TCM is considered to be a driver task, VT$TCQ, with a controller table and a TIDB. All requests are queued to the TCD request queue, except OPEN/CLOSE requests, which are queued on the terminal OPEN/CLOSE request queue (TC$OCM) for processing by VT$OCT, the Terminal Unit OPEN/CLOSE module. Because the function of VT$TCQ is limited to queuing requests, this component may not have to be modified. It should be noted that VT$TCQ also currently performs an immediate type function request for clearing I/O on a terminal and setting it down. If this has to be changed, VT$TCQ will have to be modified, otherwise, coding a TCM should not involve changes in VT$TCQ.

The main TCM executive routine in TCMEXEC is TC$CEX, which is responsible for initiating and completing TCM requests. TCMEXEC operates as an independent, multiprogrammed task and is activiated by VT$TCQ when requests are queued on a TCD, or as consequence of an expired type 3 delay, or a completion of a CCM I/O request. (NOTE: The CCM generates a pseudo interrupt by setting the event word (TBEVNT) of TCMEXEC's TIDB non-zero, when it is time-delay active.

TC$CEX is composed of three main loops. The first one checks all TCD's for any completion of active CCM requests or timeout conditions on READ request which are timeout active. The second loop checks all TCD's for requests that may be initiated and if there is one, TC$CEX does a Jump-and-Mark into the primary entry point of the appropriate TCM request processor, and this address is kept in the TCM processor table in CTBL. The third loop checks for the shortest timeout value specified for READ requests and this value is used for a type 3 delay request which suspends TCMEXEC until a CCM request completes or the time delay expires. At this point, the user must consider how TCM requests are initiated and completed by TCMEXEC and how TCM request processing programs work, because the bulk of coding a TCM lies in coding the appropriate request processors.

In general, a TCM request processing program first checks if a request can be initiated from information kept in the TCD. If it cannot be initiated because of the current status of the line or terminal, then the program should just exit and return to TC$CEX. If a TCM request can be initiated, then the program should initiate a series of one or more CCM requests to perform the required steps called for by the particular TCM request for completion. In order to initiate a CCM request, the user must first allocate memory for the CCM request block from the memory allocation pool. This is accomplished via calls on VT$GTM, the memory allocator program, through V$EXEC. When memory has been allocated, the program can build the CCM request block by calling TC$BRQ, which builds the skeleton request block, including instructions for doing a Jump-and-Set register into IOC, from information in the TCD. Other information from the TCM request can be entered into the CCM request block and the CCM request can be queued through IOC by doing an indirect jump to a location which contains the address of the CCM request block. The following is a description of how CCM request blocks are constructed.



* These values are valid only when C = 1 (Request Complete).

** Optional, since input/output may be performed directly into user's buffer or extension word is not needed.

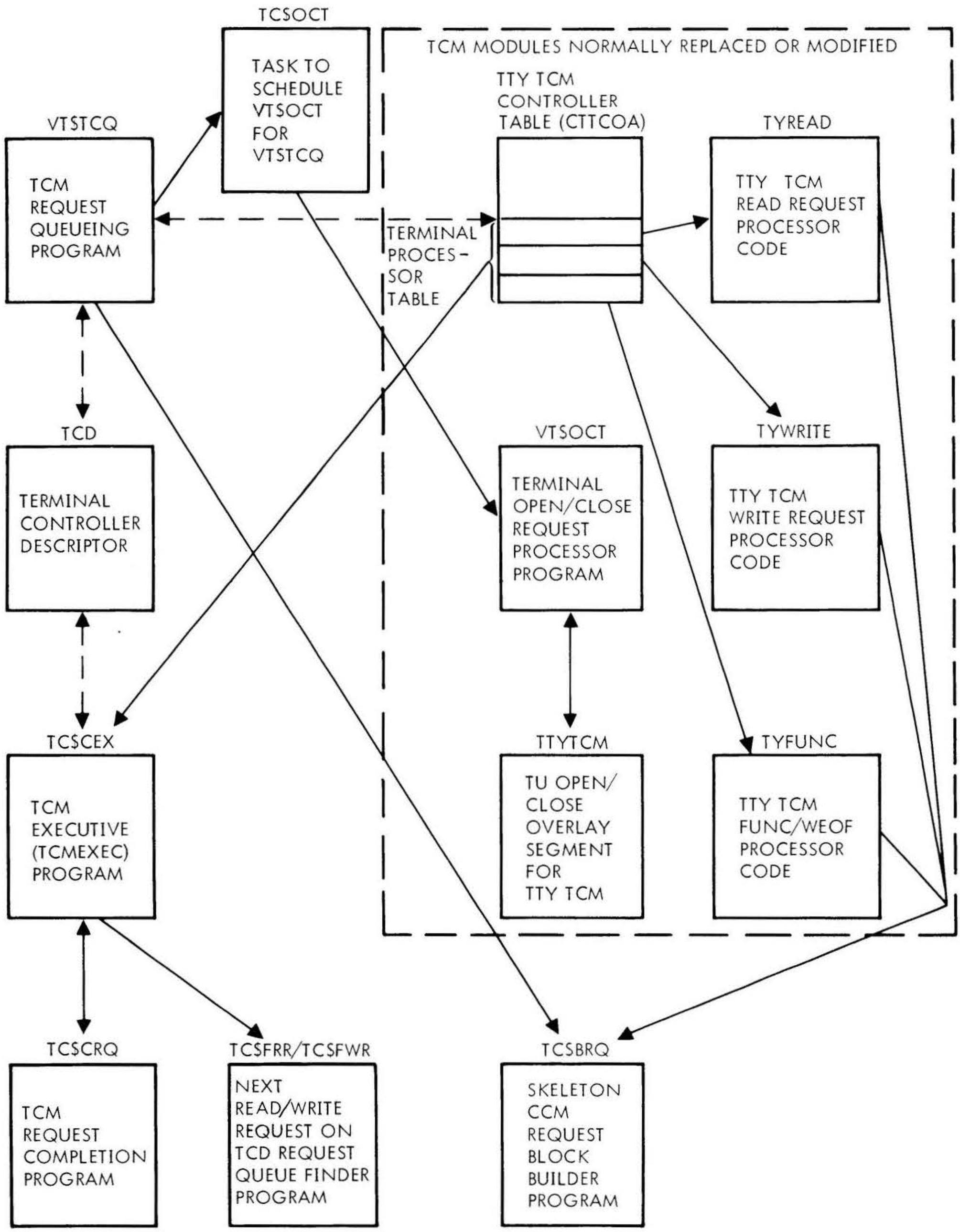


Figure 10–1. VTAM TCM and TTY TCM Modules

The TCM request processing program would normally set up the following items after the skeleton CCM request block has been built:

a. W - Wait or Immediate Return option.

b. MODE - Mode of request.

c. OP CODE - Type of request (READ, WRITE, FUNC, etc.).

d. RETURN/COMPLETION ADDRESS - Return address after IOC call. This would normally be a return address in the TCM processor program after a request is queued/completed. When the immediate return option is used, this location is also used to store a request completion address. After the program exits to TCMEXEC, control can be returned an entry point within the program from TCMEXEC (after the CCM request or a buffer, if in the buffer chain mode, has completed). This permits the completed request to be processed and further servicing or completion of the TCM request can proceed.

e. BUFFER LENGTH - Length of input/output buffer.

f. F - Word/Byte Count Flag (must be byte count if in the buffer chain mode).

g. BUFFER ADDRESS - Address of input/output buffer (address buffer in the data chain when in the buffer chain mode).

h. FUNC CODE - Function code of FUNC request.

The address of currently active CCM request blocks are stored into the following entries in the TCD:

TCRCA - Read Completion Address (Also used for FUNC).

TCWCA - Write Completion Address

When these entries are non-zero, TCMEXEC assumes that the terminal is active with CCM requests waiting to be completed. Thus, TCMEXEC can check for request completion by testing the completion bit in the CCM request block and if completed, TCMEXEC will perform a Jump-and-Mark to the completion address that was stored in word 8 of the CCM request block by the particular TCM request processing program servicing the request. When the TCM processor is reentered it would normally check for line errors by checking the detail status word returned by the CCM or the error flag (e) and completion code (cc) fields. If errors occurred then the TCM request should be completed and an appropriate error status returned. Otherwise, the TCM processor should continue request servicing or complete the TCM request by calling TC$CRQ (TCM Complete request program) and return normal completion status. It should be noted that TC$CRQ also delinks the request from the TCD request queue when it completes the request and handles error conditions like Data-Set-Ready OFF, Parity error, etc. by returning the proper error status. Lastly, memory used for CCM request blocks should be deallocated and returned to the memory pool by calling VT$PTM through V$EXEC. Then before returning to TCMEXEC, the TCM processor should clear TCRCA or TCWCA, or whatever entry is used to keep track of active CCM requests to ensure that TCMEXEC will no longer consider the TCM active with a READ, WRITE, FUNC, etc. request.

If the buffer chaining mode is specified, the completion address is called each time a buffer or a request is completed.

10.5 MODIFYING THE NETWORK DEFINITION MODULE

Modifying the NDL Processor

Additions may be made to the NDL processor by the user. In order to make these alterations, one must understand the conventions and mechanisms NDM uses to accomplish its work.

All syntactic analysis is done in PARSE, a FORTRAN subroutine. A major portion of this code was produced from a BNF notation. The original BNF syntax appears in the comment lines.

PARSE looks for particular phrases in the input stream. Each phrase is stored as a character string via DATA statements in subroutine COMPAR. PARSE requests a check for a phrase by calling COMPAR and passing the phrase number. COMPAR reflects the result of the comparison via the COMMON variable ITEST. If the phrase occurs, ITEST is set to one and the phrase is deleted from the input buffer. If the comparison fails, ITEST is set to zero.

When a phrase is found, an action is taken. Most of these actions are calls to BITSET to set fields within the control blocks. If the expected phrase does not occur and an alternative exists, the alternative is tried. If no alternatives exist, subroutine DIAG is called to produce a syntax error message, and a suitable default action is taken.

For instance below is the code within PARSE to process the STATUS clause of the TERMINAL directive. On the right are descriptive comments.

```
C     / 'STATUS" ='                  BNF statement of alternative
81    CONTINUE                       Start of alternative
      IF (ITEST.EQ.1) GO TO 83       If previous alternative true, skip this one
      CALL COMPAR (34)               Compare for 'STATUS'
      IF (ITEST.EQ.0) GO TO 83       If failed, try next alternative
      CALL COMPAR (4)                Compare for '='
      IF (TEST.EQ.0) CALL DIAG       If failed, issue message
```

(continued)

```
C     ('UP' [BITSET(TCDI(3),15,15,0)]
      CALL COMPAR (35)                  Compare for 'UP'
      IF (ITEST.EQ.0) GO TO 84          If failed, try 'DOWN'
      CALL BITSET (TCDI(3),15,15,0      Do action for 'UP'

C     /'DOWN' :[BITSET (TCDI(3),15,15,1)]
84    CONTINUE                          Start of 'DOWN' clause
      IF (ITEST.EQ. 1) GO TO 85         If 'UP' worked, skip 'DOWN'
      CALL COMPAR (36)                  Compare for 'DOWN'
      IF (ITEST.EQ.0) GO TO 86          If failed, try next alternative
      CALL BITSET (TCDI(3),15,15,1)     Do action for 'DOWN'
86    CONTINUE
85    CONTINUE
      IF (ITEST.EQ.0) CALL DIAG         If both 'UP' and 'DOWN' failed, issue
                                          message beginning of next alternative
83    CONTINUE
```

Now suppose a user wanted to alter NDL to recognize a third alternative to STATUS, for instance STATUS = MAYBE. When this is detected, PSD word 4, bit 15 is to be turned on.

First, 'MAYBE' is a new phrase and must be added to subroutine COMPAR's list of phrases. Assume that 'MAYBE' becomes string 53. The following changes would be made to COMPAR:

- replace the DIMENSION statement for STRING and POOL:

```
DIMENSION STRING (54), POOL (293)
```

- insert the following DATA cards to describe the phrase (lower case b indicates a blank within a Hollerith constant).

```
C STRING 53 5HMAYBE
  DATA STRING (54) /288/, POOL (288) /5/
  DATA POOL (289) /2HbM/, POOL (290) /2HbA/
  DATA POOL (291) /2HbY/, POOL (292) /2HbB/
  DATA POOL (293) /2HbE/
```

Then the following statements would be inserted in subroutine PARSE, following statement number 86:

```
      IF (ITEST.EQ.1) GO TO 986       If 'DOWN' worked,
                                          skip 'MAYBE'
      CALL COMPAR (53)                Compare for 'MAYBE'
      IF (ITEST.EQ.0) GO TO 986       If failed, try next
                                          alternative
      CALL BITSET (LSD(5),15,15,1)    Do action for'MAYBE'
986   CONTINUE
```

10.6 PROCEDURE TO CODE A TCM FOR VTAM

In summary, the following steps should be taken in coding a TCM for VTAM:

a. Perform an analysis of terminal requirements and line discipline for the proposed data communications network.

b. Define the structure of the terminal-oriented tables (TCD) to be used by the TCM.

c. If there exist terminal or line attributes not described by NDL, then the NDL processor may have to be modified to include these attributes.

d. Design a terminal unit OPEN/CLOSE processor overlay segment that can be called by the root segment, VT$OCT, by keying on the TCM type field of the prototype TCD. This overlay segment should build the TCD in main memory from the prototype TCD and other information built by the NDL processor during network definition.

e. Analyze the modifications to VTAM in relation to its impact on NCM, the network control module. Changes which require modifications to NCM should be avoided.

f. Design the TCM around the existing TCM Executive, TCMEXEC, components: VT$TCQ, TC$CEX, TC$CRQ. If additional services are required from TCMEXEC which are not currently provided, then the particular TCMEXEC component may have to be modified. After coding the TCM request processor programs, the TCM controller table with its TCM processor table should be built. When all these VTAM components are assembled, then a system generation to build the VTAM system should be performed and an NDL run made to define the communications network.

SECTION 11
CONTROLLING A NETWORK

11.1 INTRODUCTION

The Network Control Module (NCM) functions as an interface between the VTAM system and the VORTEX operator. The operator uses NCM for removing and adding lines and terminals to and from an on-line active data-communications network, for redirecting I/O from one terminal to another, and for listing the status of lines and terminals.

NCM operates as a foreground VORTEX task and is invoked by the operator with an OPCOM schedule request. VTAM does not need to be running to start NCM.

Directives to NCM are entered on the current OC device, and the results are reported on the OC unit. In addition to the directives provided by NCM, more extensive changes to the VTAM network are possible through the network definition language (described in section 2).

The directives in NCM are as follows:

UP	set a line/terminal on-line
DOWN	set a line/terminal off-line
REDIRECT	redirect one terminal's I/O to another
RESTORE	restore I/O to original terminal
LIST	list current status of a line/terminal
END	terminate NCM task

Many of these functions alter fundamental VTAM tables, so care should be taken in the use of NCM. For instance, if an operating VTAM terminal is DOWNed, NCM purges current I/O requests, marks the terminal down, and resets the VTAM files. This obviously could cause data for that terminal to be lost.

11.2 DIRECTIVES

11.2.1 General Format of NCM Directives

All NCM directives have the following general format:

dir, *p(1),p(2),...,p(n)* *.comment*

where **dir** is the directive name and *p(1),p(2),...(n)* is the parameter list in which individual parameters are separated with commas. The actual parameters are defined by the directive. All blanks are ignored. Equal signs are treated as commas. The period-comment field is optional.

The maximum length of a directive is 72 characters.

11.2.2 UP Directive

The UP directive causes the current status of either a logical line or terminal to be marked as on-line and available for I/O.

If a line is specified, NCM marks the prototype LSD on VTAM file VT$DFL and Physical Line Table (PLT) as UP.

If a terminal is specified, the corresponding TUID index in file VT$DFT and the Logical Terminal Table (LTT) are marked as UP.

The format is:

UP, **u(1)** , *u(2),...,u(n)*.

where each u(i) is either a terminal identifier or a logical line specifier as defined in the network definition language. A logical line specifier is a pair < c, l > , where c = CCM VORTEX logical unit number, and l = VTAM logical line number.

Any number of units (up to a total directive length of 72 characters) may be specified and will be processed in order. An error message is given, if the unit specified was not defined in NDL or is already UP.

Examples of UP Directive

Example 1:

Suppose the VORTEX CCM logical unit number is 182. Change the status of logical line 012 to on-line.

UP, 182,012

Example 2:

Vary the status of logical line 012 and terminal RM01 on-line.

UP, 182,012,RM01

11.2.3 DOWN Directive

The DOWN directive causes the current status of a terminal or logical line to be marked as off-line and not operational.

If a terminal is specified, the corresponding terminal identifier in file VT$DFT and the logical terminal table (LTT) is marked DOWN. If the specified terminal is OPEN, a FUNC (code 4) is issued to clear all I/O.

A logical line specified as a parameter to the DOWN directive causes the corresponding prototype LSD in file

VT$DFL and the Physical Line Table (PLT) to be marked DOWN. If the line is OPEN at the time, a FUNC (code 21) is issued to clear all CCM I/O requests. Then for all terminals currently OPEN on the line, a FUNC (code 4) is issued to clear all TCM I/O requests.

The DOWN directive does not close all lines and terminals associated with the downed unit. Instead, it marks the downed units RMD (prototype down), clears any current requests against the downed unit, and causes future requests to be rejected. Further, if a line is downed, all current requests for all terminals on that line are cleared. Also all future requests for these terminals on the downed line are rejected. In this manner, DOWN is the functional inverse of UP.

The form of the DOWN directive is:

DOWN, u(1),*u(2),..., u(n)*.

where each u(i) is either a terminal identifier or a logical line specifier (see section 11.2.2). Any number of units (up to a total directive length of 72 characters) may be specified in this directive and they are processed in order. If the unit specified was not defined by NDM, an error message is given. If the unit is currently DOWN, error is indicated.

Loss of data may occur, if the unit specified is OPEN.

Examples of a DOWN Directive

Example 1:

Suppose the VORTEX CCM logical unit number is 182. Set logical line 012 down.

DOWN, *182,012*

Example 2:

Set logical line 012 and terminal RM02 down.

DOWN, *182,012,RM02*

11.2.4 REDIRECT Directive

The REDIRECT directive allows the operator to substitute another terminal to receive and transmit messages. This would be useful for terminal and/or line failures.

The network control module alters the TUID index entry to point to a different prototype TCD. This changes not only the logical line for the TUID but also may change the physical hardware characteristics for the TUID.

The general form of this directive is:

REDIRECT, l(1) = **r(1)**, *l(2) = r(2),...,l(n) = r(n)*.

Each l(i) and r(i) are defined TUID's, for which r(i) replaces the l(i).

If any of the terminals specified by l(i) or r(i) were not defined by the NDL processor, an error is given in the following format:

NCnn

Any number of TUID pairs may be specified in the directive up to a total length of 72 characters. A comma may be substituted for an equal sign.

If the terminal being reassigned is OPEN at the time, it may be necessary to DOWN the terminal. Since only RMD files are altered by this directive, the reassignment takes effect when the terminal is OPENed.

Examples of REDIRECT Directives

Example 1:

Reassign I/O from terminal RM01 to terminal RM02.

```
REDIRECT, RM01 = RM02.
```

Example 2:

Terminal XRAY has failed, so shift its I/O requests to BETA.

```
REDIRECT,XRAY,BETA.
```

11.2.5 RESTORE Directive

The RESTORE directive restores terminal I/O requests to the original terminal. The TUID may have been altered by the REDIRECT directive. The format of this directive is as follows:

RESTORE, t(1), *t(2),..., t(n)*.

Each t(i) is a TUID of a terminal to be restored.

Any number of TUID's may be specified (not exceeding the total directive length of 72 characters). Each is restored in turn left to right.

Error message NC03 UNDEFINED TUID appears if any of the parameters of RESTORE had not been defined before this in NDL.

Since only RMD tables are changed by the directive, it may be necessary to DOWN the terminal. The change takes effect only when the terminal is being OPENed.

Examples of RESTORE Directives

Example 1:

Terminal DOG has been REDIRECTED. Restore its original status.

```
RESTORE,DOG.
```

Example 2:

Restore terminals REDIRECTED in section 11.2.4. example 2.

```
RESTORE,BETA,XRAY.
```

11.2.6 LIST Directive

The LIST directive lists the current status of VTAM logical lines and terminals. NCM searches the VTAM files and resident tables for information, such as UP/DOWN, OPENED/CLOSED and current assignments. A message is formatted and written to the OC device.

If no parameters are given on the directive, NCM lists the status of all defined VTAM lines and terminals. No files or tables are altered by LIST.

The format of the LIST directive is:

LIST, *u(1), u(2),..., u(n).*

each u(i) is either a TUID or a logical line specifier (see section 11.2.2) for which the status is to be listed.

In the output format of the LIST directive, the P field contains the physical line number and the T field contains the logical unit number assigned by the JCP directive.

Examples of LIST Directive

Example 1:

List the current status of terminals TTY1 and TTY2.

```
LIST,TTY1,TTY2
```

Example 2:

List the current status of terminals LA and NY and logical lines 01 and 02 both on the CCM assigned to VORTEX logical unit number 17.

```
LIST,LA,NY,17,01,17,02
```

Example 3:

List the current status of all VTAM terminals and lines.

```
LIST
```

NCM Error Codes

NC01	Syntax error
NC02	Undefined Line
NC03	Undefined TUID
NC04	I/O Error on file VT$DFL
NC05	I/O Error on file VT$DFT
NC06	Undefined CCM Number

SECTION 12
PROGRAMMING AN APPLICATION

This section presents a simple data communication example, an assembly-language program to handle inquiries from a terminal about a data base stored on a rotating-memory file. The inquiries are fixed-format messages of four-characters. The terminal handled by this program is a Teletype-compatible CRT device.

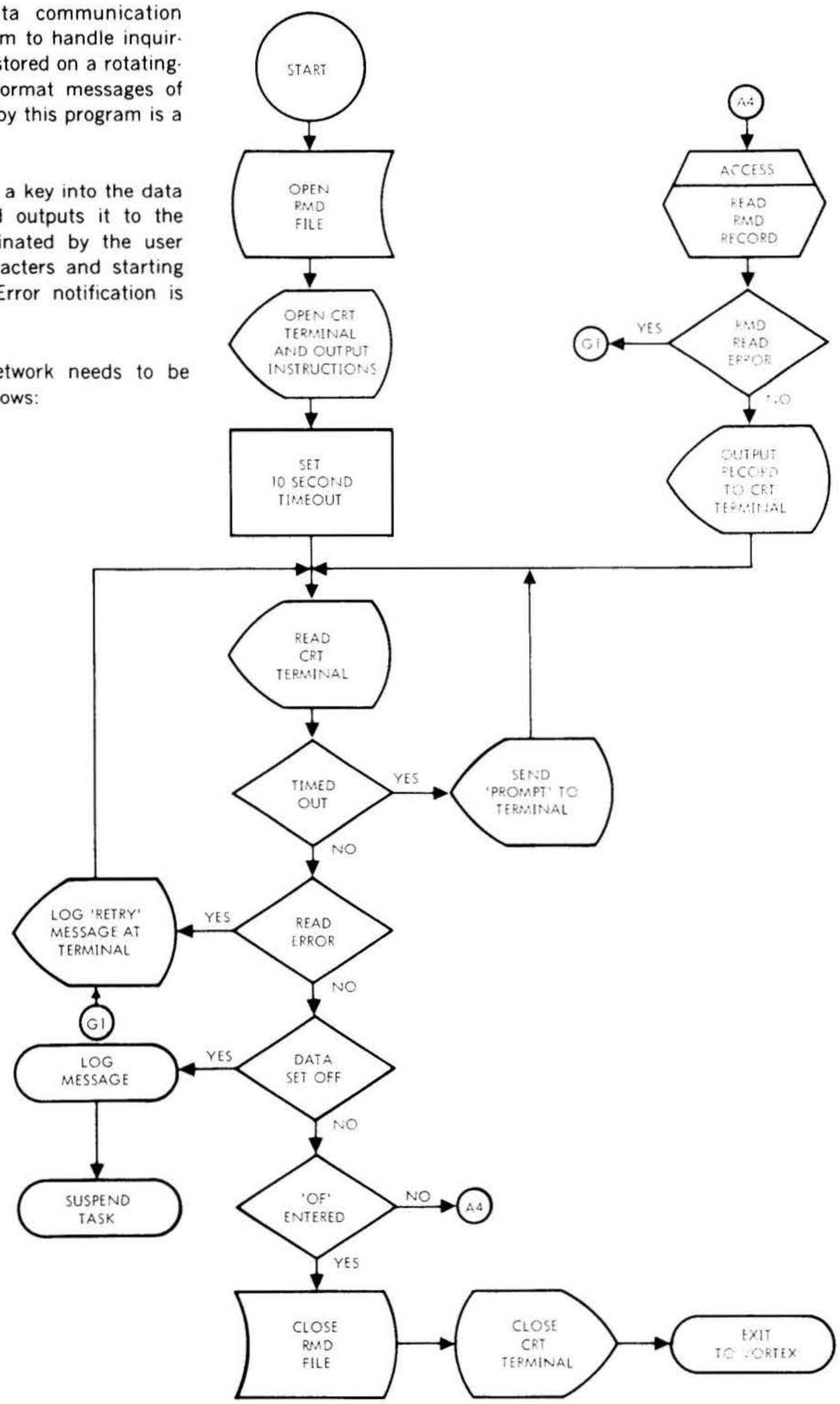
The program converts the messages to a key into the data base, reads the specified record and outputs it to the terminal. An inquiry session is terminated by the user entering "OF". Editing, deleting characters and starting over, is provided through the TCM. Error notification is provided by the program.

Before running this program the network needs to be configured with NDL statements as follows:

LINE 2:

```
        ADDRESS = 040,
        CONNECT = DIRECT,
        EOM-STOP = 0215,
        ERROR-STOP = TRUE,
        PARITY = EVEN,
        SPEED = 10,
        LINE-TYPE = HALF-DUPLEX,
        MODE = ASYNCHRONOUS,
        STATUS = UP.

TERMINAL CRT1:
        LINE = 2,
        CODE = ASCII,
        DEVICES = 1,
        ECHO = TRUE,
        PROMPT = 0207,
        TYPE = TTY1,
        UNIT = 17,
        STATUS = UP.
END.
```



VTII-1928

Figure 12-1. Flowchart of VTAM Application

```
*****************************************************************
*                                                               *
*       VTAM SAMPLE PROGRAM                                     *
*                                                               *
*       THIS PROGRAM READS A FIXED FORMAT, 4 CHARACTER MESSAGE  *
*       FROM AN ASYNCHRONOUS TTY COMPATIBLE TERMINAL.  IT THEN  *
*       CONVERTS THE MESSAGE TO A BINARY NUMBER AND USES IT AS  *
*       THE LOGICAL RECORD NUMBER TO RANDOMLY READ FROM A FILE. *
*       ONCE READ, THE RECORD IS OUTPUT TO THE TERMINAL.  ERROR *
*       NOTIFICATION AND PROGRAM TERMINATION IS ALSO PROVIDED.  *
*                                                               *
*****************************************************************
*
*       EQUATES FOR LOGICAL UNITS
*
LU1     EQU     180
LU2     EQU     186
LU3     EQU     1
*
*       BEGIN PROGRAM
*
        NAME    P0
P0      OPEN    DATAB,LU1       OPEN RMD FILE CONTAINING DATA BASE
        OPEN    TUID,LU2        OPEN TERMINAL
        WRITE   INSTR,LU2       WRITE INSTRUCTIONS TO TERMINAL
        FUNC    CRT1,LU2        SET TCM TO PROMPT AFTER 10 SECOND TIMEOUT
P1      READ    CRT1,LU2,,1     READ TERMINAL (ASCII MODE)
*                               STATUS READ AFTER COMPLETION
*
*                               ER1=I/O ERROR ROUTINE
*                               ER2=DATA SET OFF ROUTINE
*                               P1=READ TIMEOUT ROUTINE
*                               P2=LOOP ON STAT,SHOULD NEVER HAPPEN
P2      STAT    P1,ER1,ER2,P1,P2
        LDA     BUF             GOOD READ,GET FIRST 2 CHARACTERS OF MESSAGE
        SRE     OF,7,010         OF  ENTERED?
        JMP     P3              NO,THEN PROCESS MESSAGE
        CLOSE   DATAB,LU1       YES,CLOSE RMD FILE
        CLOSE   TUID,LU2        CLOSE TERMINAL
        EXIT                    RETURN TO VORTEX
P3      LDB     BUF+1           GET SECOND 2 CHARACTERS OF MESSAGE
        CALL    ACCESS          ACCESS DATA BASE
        JAN     ER1             ERROR?
        WRITE   DATAB,LU2       NO,OUTPUT RECORD TO TERMINAL
        JMP     P1              LOOP TO READ NEXT REQUEST
*
*       TERMINAL OR RMD PARITY ERROR
*
ER1     WRITE   ERMSG1,LU2      LOG ERROR MESSAGE AT TERMINAL
        JMP     P1              TRY READ AGAIN
*
*       DATA SET OFF ERROR
*
ER2     WRITE   ERMSG2,LU3      LOG ERROR TO OPERATOR
        SUSPND                  WAIT UNTIL RE-SCHEDULED OR ABORTED
*
*       ACCESS DATA BASE SUBROUTINE
*       ENTER:  A,B=RECORD NUMBER AS 4 CHARACTER ASCII NUMBER
*       EXIT:   DATAB=RECORD
*               A=+ (NO ERROR)
*               A=- (ERROR)
*
ACCESS DATA     0
        CALL    ASBI            CONVERT INPUT TO BINARY
```

```
        STB     RECNO           USE AS RANDOM ACCESS RECORD NUMBER IN FCB
AC1     READ    DATAB,LU1       READ RANDOM RECORD
        LDAE    AC1+2           GET I/O STATUS WORD
        LRLA    7               POSITION ERROR BIT TO SIGN
        JMP*    ACCESS          EXIT
*
*       CONVERT ASCII TO BINARY SUBROUTINE
*       ENTER:  A,B=4 CHARACTER ASCII NUMBER
*       EXIT:   B=BINARY EQUIVALENT
*
ASBI    DATA    0
        STA     A1              SAVE HIGH-ORDER 2 DIGITS (D1-2)
        TBA
        ANA     BM17            ISOLATE D4
        TAX                     SAVE
        LLRL    8
        ANA     BM17            ISOLATE D3
        TAB
        TXA
        MUL     TEN             D3*10+D4*1
        TBX                     SAVE
        LDA     A1
        ANA     BM17            ISOLATE D2
        TAB
        TXA
        MULI    0144            D2*100+D3*10+D4*1
        TBX                     SAVE
        LDA     A1
        LRLA    8
        ANA     BM17            ISOLATE D1
        TAB
        TXA
        MULI    01750           D1*1000+D2*100+D3*10+D4*1
        JMP*    ASBI            EXIT
A1      DATA    0
*
*       EQUATES FOR VORTEX LOWER MEMORY CONSTANTS
*
TEN     EQU     0471
BM17    EQU     0472
*
*       DATA CONTROL BLOCKS AND BUFFERS
*
DATAB   FCB     36,DATABF,,,`FI',`LE',`01'
DATABF  BSS     36
RECNO   EQU     DATAB+3
*
INSTR   DCB     33,MSG0
MSG0    DATA    `DATA BASE INQUIRY. TYPE 4 DIGIT KEY TO ACCESS, "OF"'
        DATA    `TO TERMINATE.'
*
TUID    DCB     `CR',`T1'
CRT1    DCB     2,BUF,05003
BUF     BSS     2
*
ERMSG1  DCB     9,MSG1
MSG1    DATA    `I/O ERROR, RETRY.'
*
ERMSG2  DCB     7,MSG2
MSG2    DATA    `DATA SET OFF.'
OF      DATA    `OF'
        END     P0
```

SECTION 13
CONFIGURING A VTAM SYSTEM

13.1 INTRODUCTION

The procedure for system generation of a system with VTAM is the same as that for VORTEX with the additional steps described in this section. Steps a, b, and c are required only if more than four DCM's are used or if modifications are needed to the standard VTAM system. Refer to the following sections for details. These additional procedures for VTAM are:

a. Modifying VTAM CCM tables and adding controller tables with installation dependent parameters.

b. Adding TDF cards and binary decks for VTAM CCM.

c. Adding TDF cards and binary decks for TTY TCM.

d. Reserving memory for DCM's control words (with MRY directive and DEF directives).

e. Defining data communications multiplexors in peripheral architecture (with EQP directive).

f. Defining interrupt structure required by DCM (with PIM directive).

g. Associating logical unit numbers and names with physical devices (with ASN directive).

h. Loading ancillary VTAM system modules (OPEN, CLOSE, NDM, and NCM) subsequent to VORTEX system generation.

13.2 MODIFYING VTAM CCM TABLES AND ADDING CONTROLLER TABLES

13.2.1 CCM Tables

If the SYSGEN EQP directives specify a TC or MX device, SYSGEN will build the VTAM data module CC$TLB. This module contains a number of variable parameters, each of which has a default value, but which can be redefined by the SYSGEN DEF directive. The names and default values of these parameters are:

The names and default values of these parameters are:

Name	Default Value
CBSIZE	15
NUMLL	20
NUMENx	64
BCTNTx	64
NULEL	32
NUTEL	10

where x refers to the DCM number

Functions of these parameters are described below.

CBSIZE is the number of 2-word entries required for the circular interrupt buffer, which must be large enough to support the maximum number of DCM interrupts that can occur simultaneously. The number of entries needed depends upon the maximum number of active lines at any time.

A value of half of the number of active lines may be adequate in most cases. A more exact determination requires an analysis of the specific communications system being generated along with the application of queuing theory.

NUMLL is the number of logical lines. The default value is 20.

NUMENx specifies the number of physical lines for the DCM MXxA. One physical line table is constructed for each DCM supported by VTAM, as defined by the SYSGEN directive:

EQP,MXxA,...

where x is the DCM number
The format of the corresponding physical line table is as follows:

```
        EXT     NUMENx
        NAME    C52PLx
C52LPx  DATA    NUMENx
        DUP     NUMENx
        DATA    0
```

The multiplexor equipment table provides the means to obtain the controller table address for an interrupting DCM. The structure of the multiplexor equipment table is:

```
CC$MET  DATA    NOMUXS
        DATA    CTMX0A
        DATA    CTMX1A
        .
        .
        .
        DATA    CTMXnA
```

NOMUXS, is the number of entries in the table. NOMUXS equals n + 1 where n is the largest DCM number if the SYSGEN EQP directives (EQP, MXnA,...).

BCTNTx is the number of lines using buffer chaining for DCM MXxA. One buffer chain table (BCT) is constructed for each DCM supported by VTAM, when buffer chaining mode of input is used, as defined by the SYSGEN directive:

EQP,MXxA,...

Each entry corresponds to one physical line number which, when in use, contains a chain header address to that line. The format for a buffer chain table is as follows:

```
                EXT     BCTNTx
                NAME    BCTXxA
BCTXxA          DATA    BCTNTx
                DUP     BCTNTx
                DATA    0
```

NUTEL is the number of opened terminals. The size of the VTAM dynamic memory pool is determined by the number of lines and terminals to be open with current I/O requests. Two pool elements are assigned for each terminal with active I/O requests at one time, and one pool element is assigned for each line opened at any one time. NULEL is the number of opened lines.

13.2.2 Controller Table

A CCM controller table must be provided for each DCM in the system. The VORTEX SGL contains controller tables for four DCM's. If more controller tables are required than are furnished with the SGL, their source modules can be created by using the controller table CTMX0A as a model.

Names of DCM controller tables must be in the form CTMXnA, where n is the controller number. CTMX0A is the name of the controller table for the first DCM, CTMX1A is the name of the controller table for the second DCM, and so forth.

Before assembly, the following changes (see the assembly listing for CTMX0A) must be made to the controller table CTMX0A:

a. Replace the controller talbe name in the NAME directive and the following EQU with the name of the controller table being assembled. For DCM 4 (the 5th DCM) this would be:

```
        NAME    CTMX4A
CTMX4A  EQU     *
```

b. Change the TIDB address, TBMXnA (word 00 CTIDB), to reflect the proper controller number. The changes for DCM 4 would be as follows:

```
        EXT     TBMX4A
        DATA    TBMX4A
```

c. Change the DEVICE ADDRESS, (word 06 CTDVA), to reflect the proper controller number in the same manner as b above.

d. Change the BIC FLAG TABLE ADDRESS, IBMXnA (word 011 CTBIC), to reflect the proper controller number in the same manner as b above.

e. If the controller table is not for DCM 0 insert after the comment:

```
* START OF DEVICE MANAGEMENT TABLE
```

an EQU which equates the symbol CTMX0A to the name of the controller table as follows:

```
* CTMX0A EQU CTMXnA
```

f. Change the LCW BASE ADDRESS, V$LCWn (word 021), to reflect the proper controller number in the same manner as b above.

g. Every VTAM system should make use of a CL flag named V$POLL. If a modified binary synchronous communications line adapter is used, the SYSGEN directive DEF,V$POLL,1 must be included. In all other cases, the SYSGEN directive DEF,V$POLL,O must be used.

After assembly of these additional controller tables the object programs are added to the system by SMAIN or SYSGEN. Note that in VORTEX II, they must be added before the SGL control record CTL,21.

13.3 ADDING TDF RECORDS FOR VTAM CCM's

A TDF record must be provided for each DCM in the system. The standard SGL supports four TCM's. If more TCM's are used the user must provide the additional TDF records. In VORTEX II, all TDF records must be added before the SGL control record CTL,21.

13.4 ADDING TDF RECORDS FOR TCM (TTY)

A TDF records must be provided for each TCM in the system. The standard SGL supports four TCM's. If more TCM's are used the user must provide the additional TDF records. In VORTEX II, all TDF records must be added before the SGL control record CTL,21.

13.5 RESERVING MEMORY

The memory parameter on the MRY directive must be set to reflect the DCM's usage a 512-word memory page for hardware control words. This page of memory must start at a multiple of 512 words, i.e. 074000, 075000 etc.

Example (VORTEX):

```
MRY, 074777,0200
```

Example (VORTEX II):

```
MRY, 074777,0200,64
```

Reserve the highest page available to VORTEX (075000 to 075777) when AID II and BLD are memory resident (AID II starts at 076000 in a 32K word memory configuration).

The LCW address for each DCM is defined using the SYSGEN directive DEF. The format for this directive as follows:

```
DEF,V$LCWn,xxxxxx
```

where n is the DCM number and xxxxxx is the LCW address in octal

For example: In a 32K system DCM 0 is wired for an LCW address of 075000. The DEF directive would be:

```
DEF,V$LCW0,075000
```

13.6 DEFINING PERIPHERAL ARCHITECTURE

An EQP directive must be made for each DCM and each TCM.

An EQP card must be present for each DCM in the system. The format for the equipment name field is:

```
MXnA
```

where n is a single numeric character.

Example:

```
EQP,MX0A,074,1,0,0
```

MX0A is the mnemonic for the first DCM in the system, 074 is its device address, 1 is the number of peripheral units (always set to 1). The last two parameters must be set to zero.

For a TCM the format of the name for the terminal control module is TCnA.

Where n is a single numeric character

Example

```
EQP,TC0A,00,1,0,0
```

13.7 DEFINING INTERRUPT STRUCTURE

For each EQP card defining a DCM six PIM directives are needed to define the six DCM interrupts.

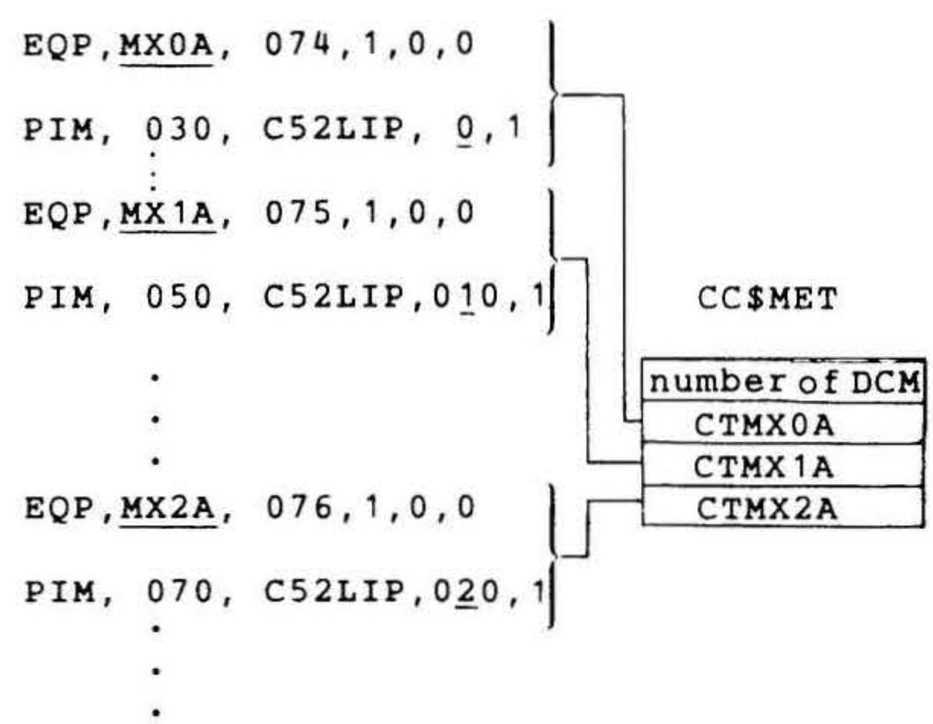
The PIM directives for a DCM define directly connected interrupts. The names of the programs servicing the directly connected interrupts are in table 13-1. For VORTEX I, specify direct option 1. For VORTEX II, specify direct connect option 2.

Table 13-1. Direct Connect Interrupts

Event Word Value	Interrupt Description	Directly Connected Interrupt Servicing Routine Name
0X0	input byte count = 0	C52LIP
0X1	output byte count = 0	C52LIP
0X2	line error	C52LIP
0X4	control character detected	C52LIP
0X3	status change	C52LIP
0X5	control	C52CIH

The event word entry in the PIM directive is taken from table 13-1, where X is the number of the DCM being described in the PIM directive. For example in a system using only one DCM, X = 0 in all six PIM cards. In a system using two DCM's the first DCM would be described by six PIM directives with X = 0, and the second by six PIM directives with X = 1.

There is a one-to-one relationship between the controller table name generated by the EQP directive, the relative position of that controller table's name in the table CC$MET and the value X as shown in the following example:



The controller table name generated by the EQP directive must be used when assembling the controller table and will be used in the construction of CC$MET. The value X will be the ordinal of the controller tables address in CC$MET.

13.8 ASSIGN LOGICAL UNITS TO PHYSICAL DEVICES

The ASN directive associates a logical unit number (1 through 100 or 107 through 255) which can be followed by an optional two-character logical unit name (e.g., 107:Y7) with a four-character physical-device name such as TCnn or MXnn, where n is a single numeric character. Note that a different TCM logical unit number is required for every terminal that is open at the same time.

EXAMPLE:

```
ASN, 26 = MX00
ASN, 184 = TC00
```

13.9 LOADING ANCILLARY VTAM MODULES

Jobs for loading OPEN, CLOSE, NDM, NCM and the FORTRAN run-time modules to support terminal open and close are provided with the VTAM release material. These jobs are run from the SI logical unit and provide the operator with any instructions necessary for their execution.

The job loading ancillary VTAM modules is organized into two parts, separated by an end-of-file record. The first part must be run for all types of VTAM system configurations, with or without TCMs. The second part, which loads the terminal open/close task, should only be run when a TCM is included in the VTAM configuration. The second part is also terminated by an end-of-file record.

NOTE: Prior to the loading of NCM in the first part of loading ancillary modules, there is a job to enter the external names: VT$LTT and TC$TCD into the OM library. These external names are needed for the load module generation of NCM. If the VTAM system was generated with a TCM, these names would be in the CL library. For systems with CCMs only, these names must be entered into OM as dummy entry points. If these names are already in CL, the entries in OM may be deleted.

13.10 VTAM MEMORY REQUIREMENTS

VTAM requires the following amounts of memory:

CCM:

Components	3200 words
Line Tables	17 words (18 words/line if buffer chaining is used)

DCM Multiplexor 512 words/multiplexor

TCM:

Components	2600 words
Terminal Tables	17 words/terminal

APPENDIX A
TELETYPE AND CRT CHARACTER CODES

Character	Internal ASCII (Octal)	Character	Internal ASCII (Octal)
0	260	R	322
1	261	S	323
2	262	T	324
3	263	U	325
4	264	V	326
5	265	W	327
6	266	X	330
7	267	Y	331
8	270	Z	332
9	271	(blank)	240
A	301	!	241
B	302	"	242
C	303	#	243
D	304	$	244
E	305	%	245
F	306	&	246
G	307	'	247
H	310	(	250
I	311	)	251
J	312	*	252
K	313	+	253
L	314	,	254
M	315	–	255
N	316	.	256
O	317	/	257
P	320	:	272
Q	321	;	273
<	274	FORM	214
=	275	RETURN	215
>	276	SO	216
?	277	SI	217
@	300	DCO	220
....	333	X-ON	221
....	334	TAPE AUX	....
....	335	ON	222
↑	336	X-OFF	223
←	337	TAPE OFF	....
RUBOUT	377	AUX	224
NUL	200	ERROR	225
SOM	201	SYNC	226
EOA	202	LEM	227
EOM	203	S0	230
EOT	204	S1	231
WRU	205	S2	232
RU	206	S3	233
BEL	207	S4	234
FE	210	S5	235
H TAB	211	S6	236
LINE FEED	212	S7	237
V TAB	213		

APPENDIX B
EBCDIC AND ASCII CHARACTER ASSIGNMENTS

Character	EBCDIC (Hex)	ASCII (Hex)	Character	EBCDIC (Hex)	ASCII (Hex)
A	C1	41	@	7C	40
B	C2	42	(	4D	28
C	C3	43	)	5D	29
D	C4	44	–	6D	5F
E	C5	45	'	7D	27
F	C6	46	+	4E	2B
G	C7	47	;	5E	3B
H	C8	48	>	6E	3E
I	C9	49	=	7E	3D
J	D1	4A	|	4F	
K	D2	4B	¬	5F	5E
L	D3	4C	?	6F	3F
M	D4	4D	"	7F	22
N	D5	4E	¦	C0	7B
O	D6	4F	¦	D0	7D
P	D7	50	\	E0	5C
Q	D8	51	~	A1	7E
R	D9	52	\	79	60
S	E2	53	:	6A	7C
T	E3	54	[		5B
U	E4	55	]		5D
V	E5	56	BEL	2F	07
W	E6	57	BS	16	08
X	E7	58	BYP	24	
Y	E8	59	CAN	18	18
Z	E9	5A	CC	1A	
a	81	61	CR	0D	0D
b	82	62	DC1	11	11
c	83	63	DC2	12	12
d	84	64	DC3	13	13
e	85	65	DC4	3C	14
f	86	66	DEL	07	7F
g	87	67	DLE	10	10
h	88	68	DS	20	
i	89	69	EM	19	19
j	91	6A	ENQ	2D	05
k	92	6B	EOB	26	
l	93	6C	EOT	37	04
m	94	6D	ESC	27	1B
n	95	6E	ETB	26	17
o	96	6F	ETX	03	03
p	97	70	FF	0C	0C
q	98	71	FS	22	1C
r	99	72	GS		10
s	A2	73	HT	05	09
t	A3	74	IFS	1C	
u	A4	75	IGS	1D	
v	A5	76	ILS	17	
w	A6	77	IRS	1E	
x	A7	78	IUS	1F	
y	A8	79	LC	06	
z	A9	7A	LF	25	0A
0	F0	30	NAK	3D	15
1	F1	31	NC	15	

EBCDIC AND ASCII CHARACTER ASSIGNMENTS

Character	EBCDIC (Hex)	ASCII (Hex)	Character	EBCDIC (Hex)	ASCII (Hex)
2	F2	32	NUL	00	00
3	F3	33	PF	04	
4	F4	34	PN	34	
5	F5	35	PRE	27	
6	F6	36	RES	14	
7	F7	37	RLF	09	
8	F8	38	RS	35	1E
9	F9	39	SI	0F	0F
&	50	50	SM	2A	
-	60	2D	SMM	0A	
/	61	2F	SO	0E	0E
$	5B	24	SOH	01	01
¢	4A		SOS	21	
!	5A	21	Space	40	20
:	7A	3A	STX	02	02
#	7B	23	SUB	3F	1A
,	6B	2C	SYN	32	16
.	4B	2E	UC	36	
<	4C	3C	US		1F
*	5C	2A	VT	0B	0B
%	6C	25			

INDEX

Active chain, 7-2
Affirmative acknowledgment (ACK0/ACK1), 8-5

Bibliography, 1-4
Binary synchronous communication (BSC), 8-1
Block-check character (BCC), 8-3
Buffer chaining, 7-1

CCM level, programming at the, 6-1
CCM tables, 13-1
Chain, active, 7-2
Chain header (CHR), 7-2
Chaining, buffer, 7-1
Character, block-check (BCC), 8-3
Character codes
  ASCII, B-1
  CRT, A-1
  EBCDIC, B-1
  Teletype, A-1
CHR and IBH, relationship of, 7-5
CLOSE macro, 4-2, 6-3
  error indications, 4-3
  example, 4-3, 6-3
Communication, binary synchronous (BSC), 8-1
Communications controller module (CCM) 1-1
Configuration, expanded, 1-3
Configuring a VTAM system, 13-1
Continue timeout, 8-9
Control
  ACK0/ACK1, 8-5
  characters, 8-4
  DLE, 8-5
  ENQ, 8-5
  EOT, 8-5
  ETB, 8-4
  ETX, 8-5
  ITB, 8-4
  NAK, 8-5
  RVI, 8-6
  SOH, 8-4
  station, 8-2
  STX, 8-4
  SYN, 8-4
  TTD, 8-6
  WACK, 8-5
Controller table, CCM, 13-2
Controller table (CTBL), TCM, 10-2
Cyclic-redundancy check (CRC), 8-3

Data link, 8-1
  control, 8-4
  escape (DLE), 8-5
  multipoint, 8-1
  operation, 8-2
Dial up operation, 8-8
Directives, 11-1
Disconnect timeout, 8-10
DLE, control, 8-5
Double pointer queue, 7-1
DOWN directive, 11-1

End-of-text (ETX), 8-5
End-of-transmission block (ETB), 8-4
End-of-transmission (EOT), 8-5
END statement, NDL, 2-5
ENQ, control, 8-5
Enquiry (ENQ), 8-5
EOT, control, 8-5
Error checking, 8-3
Error indications
  CLOSE macro, 4-3
  OPEN macro, 4-2
  VTAM macros, 3-2
ETB, control, 8-4
ETX, control, 8-5
Expanded configuration, 1-3

Format, chain header, 7-2
Format, IBH, 7-2
Front pointer, 7-1
FUNC macro, 5-4, 6-4
Function codes, 6-4

GETQ, 7-1
  example, 7-4

Header, chain (CHR), 7-2

Initialization procedure, 8-6
Interface block header (IBH), 7-2
Intermediate block (ITB), 8-4
Introduction, 1-1

LCB macro, 6-1
LCB status, 3-2
Leased line, 8-1

Line control block (LCB), 3-1
Line service descriptor, prototype, 2-6
LINE statement, 2-1
  attributes, 2-2
  attribute defaults, 2-4
  examples, 2-4
List directive, 11-3
Longitudinal-redundancy check (LRC), 8-3

Macro,
  CLOSE, 4-2, 6-3
  FUNC, 5-4, 6-4
  LCB, 6-1
  OPEN, 4-1, 6-3
  READ, 5-1, 6-3
  STAT, 5-3, 6-7
  WEOF, 5-6
  WRITE, 5-2, 6-3
Managing buffers, 9-1
Memory allocation routines, 9-1
Message
  blocks, 8-3
  format, 8-2, 8-6
  transfer procedure, 8-8
Minimum configuration, 1-3
Modifying the NDL processor, 10-5
Modifying the NDM, 10-5

NDL processor, modifying the, 10-5
NDM, modifying the, 10-5
Negative acknowledgment (NAK), 8-5
Network definition language (NDL) statement, 2-1
Network control module (NCM), 1-1
Network definition module files, 2-6
Network definition module (NDM), 2-1
Network definition module output, 2-6

OPEN macro, 4-1, 6-3
  error indications, 4-2
  examples, 4-2, 6-3
Opening and closing terminals and lines, 4-1

Pad characters, 8-9
Pad format check, 8-4
Point-to-point operation, 8-6
Polling, 8-2
Processor table, TCM, 10-3
Programming an application, 12-1
Programming at the CCM level, 6-1
  CLOSE macro, 6-3
  FUNC macro, 6-4
  LCB macro, 6-1
  OPEN macro, 6-3
  READ macro, 6-3
  STAT macro, 6-7
  WRITE macro, 6-3
Programming at the TCM level, 5-1
  FUNC macro, 5-4
  READ macro, 5-1
  STAT macro, 5-3
  WEOF macro, 5-6
  WRITE macro, 5-2
Prototype line service descriptor, 2-7
Prototype terminal control descriptor, 2-7
PUTQ, 7-1
  example, 7-3

Queuing procedure, 7-1

READ macro, 5-1, 6-3
Rear pointer, 7-1
Receive timeout, 8-10
Redirect directive, 11-2
Reverse interrupt (RVI), 8-6
Reserving memory, 13-2
Reset function, 7-4
Restore directive, 11-2

Selection, 8-2
Set function, 7-4
Start-of-heading (SOH), 8-4
Start-of-text (STX), 8-4
STAT macro, 5-3, 6-7
Structure of VTAM, 1-2
Switched network, 8-1
Synchronous idle (SYN), 8-4
System flow, 1-1

Tables
  CCM controller, 13-1
  TCM controller, 10-2
  used by TCM, 10-1
TCM components, 10-3
TCM functions, 10-2
Temporary text delay (TTD), 8-5
Terminal control descriptor, prototype, 2-7
Terminal control module (TCM), 1-1
Terminal identifier block (TIB), 2-7
TERMINAL statement, 2-4
  attribute, 2-5
Termination procedure, 8-8
Timeouts, 8-9
Transmission codes, 8-1
Transmission and recovery procedures, 8-10

Transmit timeout, 8-9
Transparent mode, 8-8

UP directive, 11-1
Using VTAM macros, 3-1

VT$BMT, 9-1
VT$GTM, 9-2
VT$PTM, 9-2

Wait-before-acknowledgment (WACK) 8-5
WEOF macro, 5-6
WRITE macro, 5-2, 6-3