USER'S GUIDE

Bull

DATA COMMUNICATIONS

DPS 7 GCOS

Communications Processing Facility
Volume 3 : MCS

Software

Subject : This manual deals with all the programming aspects of communications in MCS COBOL and GPL.
It describes how MCS processes data, how the system implements the line procedures and how to program terminals. It also deals with the QMAINT utility.

Special Instructions : This manual belongs to a set of three volumes which deal with the subject «Communications Processing Facility», namely,
.47 A2 01UC Communications Overview
.47 A2 02UC Network Generation
.47 A2 03UC MCS User's Guide
Change bars indicate technical modifications or additions, while asterisks indicate deletions.

Software Supported : GCOS 7-LS, GCOS 7-MS Release V 1

Date : August 1984

Bull **CEDOC - CELOG**

Boîte Postale 110 Parc Industriel d'INCARVILLE
27100 Ensemble Urbain du VAUDREUIL - FRANCE

47 A2 03UC REV0

Dépot légal
3ème trimestre 1984

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this manual are invited. A form is provided at the end of this manual for this purpose.

PREFACE

This manual is the third of three volumes dealing with the subject "Communications Processing Facility".

The MCS User Guide deals with all programming aspects of communications in GPL and MCS COBOL, which includes not only the communications elements but also the use of run-time packages. It treats the system implementation of the line procedures and gives examples of how to program terminals. The manual deals with the utility QMAINT, which enables the user to perform maintenance on disk and memory queues defined for the network at CNC generation.

In the three volumes of "Communications Processing Facility", the term "64/DPS 7" is synonymous with "DPS 7", the prefix "64" being a "carry-over" from previous releases. The keyword "HL64" designates the DPS 7 as a CPU-terminal in the URP local network and appears in the CNC declaration described in the Network Generation Manual.

The term "MAM" (Message Access Method) is used only in the context of MAM as a system module of GCOS 7, such as MAM Error Messages in the Job Occurrence Report. Otherwise, the term "MCS" (Message Control System) which groups MAM with QMON, is used to qualify user-defined communications applications.

For the current release, synchronous links for terminals of the VIP line procedure are directly supported via a URP packet link over the TRANSPAC secondary network. These synchronous links are provided by
. DCU7010 convertor for VIP-type terminals : VIP7001 / 7700 / 7760, TTS7800 & TTU8221
. TCU7022 / 7043 terminal concentrators for QUESTAR terminals : DKU7007 / 7107 / 7211.

TRANSPAC refers specifically to the public data network in France, the use of which is contracted by PTT subscription.

This manual is intended for the systems engineer and programmer/analyst for specifying and writing an MCS application for DPS 7 communications systems. The DPS 7 communications system is set up and tailored by the use of the CNC utility, see the Network Generation Manual.

Section I introduces the essential user-visible interface with the communications network. This section which leads on to the user-facilities provided by MCS, begins with a synopsis of "Communications Overview" which also serves as a reference to Distributed Systems Architecture as implemented by the DPS 7.

Section II describes MCS in terms of a user-queue access method shared at system level allowing the exchange of messages to and from the terminals. It explains the structure of the MCS load-module and the means of linking mono- and multi-process load-modules. It also gives the run-time JCL for the MCS application.

Section III deals with MCS data processing, in particular, the symbolic representation of data, and the various ways that MCS deals with data formats. It also gives details of representing data in the form of graphic symbols that take into account national language options.

Section IV describes the dynamics of establishing the connection between users, represented by terminals and applications, the connection interface between local and remote users being handled by MCS. It deals with the various types of connections available over the DPS 7 network.

Section V deals with the line procedures supporting transmission protocols for terminals available to MCS. It describes these procedures implemented at system level and, apart from BSC2780, are not user visible. In the case of BSC2780, certain reserved control codes must not be used by the application.

Section VI gives examples of how to program terminals according to their functional characteristics and line procedures. This information concerns the programming interface for both MCS as well as TDS. The difference in these two subsystems is that symbolic representation of data is not available for TDS.

Section VII deals with QMAINT through which the user maintains the queues declared at network generation. These disk and memory queues can be systematically updated, that is, purged or redefined, at the end of the communications session. In addition, QMAINT allows the listing of queue status within the network.

Section VIII describes the events occurring during a communications session in terms of the parts played by the user and the system. Dynamics of communications is a detailed account of the network environment in which the various software components interact to enable message exchange and allocation of resources.

Appendix A gives an example of an MCS application written in MCS COBOL and GPL, testing the TWA option, in which the operator must key in the required reply.

Appendix B lists the QMAINT sysout report which enables the user to ensure that maintenance functions specified on the queues have been executed.

Appendix C lists and explains MAM error messages in the Job Occurrence Report, QMAINT error messages in the sysout report and the JOR, and the "return-codes".

Appendix D lists and explains the communications status keys, in terms of the conditions in which they occur for the five communications verbs.

The other two volumes dealing with "Communications Processing Facility" are,
. 47A2 01UC Communications Overview
. 47A2 02UC Network Generation.

The Communications Overview Manual describes how DPS 7 network processing implements DSA techniques. It treats both primary and secondary networks and explains how both are supported by GCOS 7 communications components. A description is also given on the TRANSPAC X.25 link.

The Network Generation Manual deals with the Communications Network Configurator (CNC) utility and is a reference document for the CNC commands used to generate the primary, secondary and TRANSPAC networks. Until the current release, the only primary network supported was the DPS 7 functioning as a "host" accessing its remote systems through the intermediary of its "front-end" system, the DN7100. For the current release, the primary network configuration has been extended to the DPS 7 functioning as "satellite" thereby directly accessing its remote systems.

The following publications give further details to the topics mentioned in this manual,

- For DPS 7 implementation of DSA concepts in a networking environment,
  - 47A2 01UC Communications Overview
- For a reference to the CNC commands when generating the DPS 7 network,
  - 47A2 02UC Network Generation
- For coding MCS applications,
  - 47A2 35UL GPL Language Reference Manual
  - 47A2 36UL GPL User Guide
  - 47A2 37UL GPL System Primitives
  - 47A2 01UL COBOL 74 Language Reference Manual
  - 47A2 02UL COBOL 74 User Guide
- For information on linking compile units,
  - 47A2 01UP LIBMAINT Reference Manual
  - 47A2 02UP LIBMAINT User Guide
  - 47A2 10UP Linker User Guide
- For file allocation when using disk queues,
  - 47A2 05UF Data Management Utilities User Guide
- For $QASSIGN and run-time JCL,
  - 47A2 11UJ JCL Reference Manual
  - 47A2 12UJ JCL User Guide
- For GCOS 7 communications operator interface affecting transmission modes,
  - 47A2 04UC Terminal Operations
  - 47A2 05UC Network Control Terminal Operations
- For GCOS 7 console operator interface for modifying scheduling,
  - 47A2 01UU System Operator's Guide
- For a description of catalog access rights,
  - 47A2 01US System Administrator's Manual
- For general information on DPS 7 installations,
  - 30A1 8265 Functional Characteristics for DPS 7/x0
  - 93A1 8695 Functional Characteristics for DPS 7/x5
  - 47A2 04UG System Overview

° For a résumé of all aspects of networking applicable to the DPS 7,

  . 47A2 09UC Telecommunications Reference Card

° For formatting the DKU7007, DKU7107, DKU7211, VIP7700 (VIP7001 and VIP7700), VIP7760 and IBM3270 screens in a TDS environment,

  . 47A2 10UT TDS/FORMS User Guide

Suggestions and criticisms concerning the form, content and purpose of this manual are invited.

A Technical Publications Remarks Form is included at the end of the manual for this purpose.

Each section of this document is structured according to the heading hierarchy shown below.

Each heading indicates the relative level of the text which follows it.

Level	Heading Format
1 (highest)	ALL CAPITAL LETTERS, UNDERLINED
2	Initial Capital Letters, Underlined
3	ALL CAPITALS, NOT UNDERLINED
4	Initial Capital Letters, Not Underlined
5 (lowest)	ALL CAPITAL LETTERS FOLLOWED BY COLON : text begins on the same line

CONTENTS

Section II (continued)

Section II (continued)

Section III (continued)

SECTION I

INTRODUCTION

MCS applications, written in either MCS COBOL or GPL, require access through queues to the communications network. These queues are defined by the user.

In this respect, such user-defined applications differ from the communications services, collectively termed VCAM subsystems. VCAM allows direct communication between these subsystems and terminals, and between the subsystems themselves. The user is not concerned with queues.

The user interface to the communications network is described in terms of

- networking environment
- communications components
- MCS
- QMAINT.

ACCESS TO MCS OVER FNPS/DN7100

The DN7100 software release concurrent with Release V1 of GCOS7 is DNS B2 (V2.6). Configurable connections over the DN7100 supported for the current release are as follows.

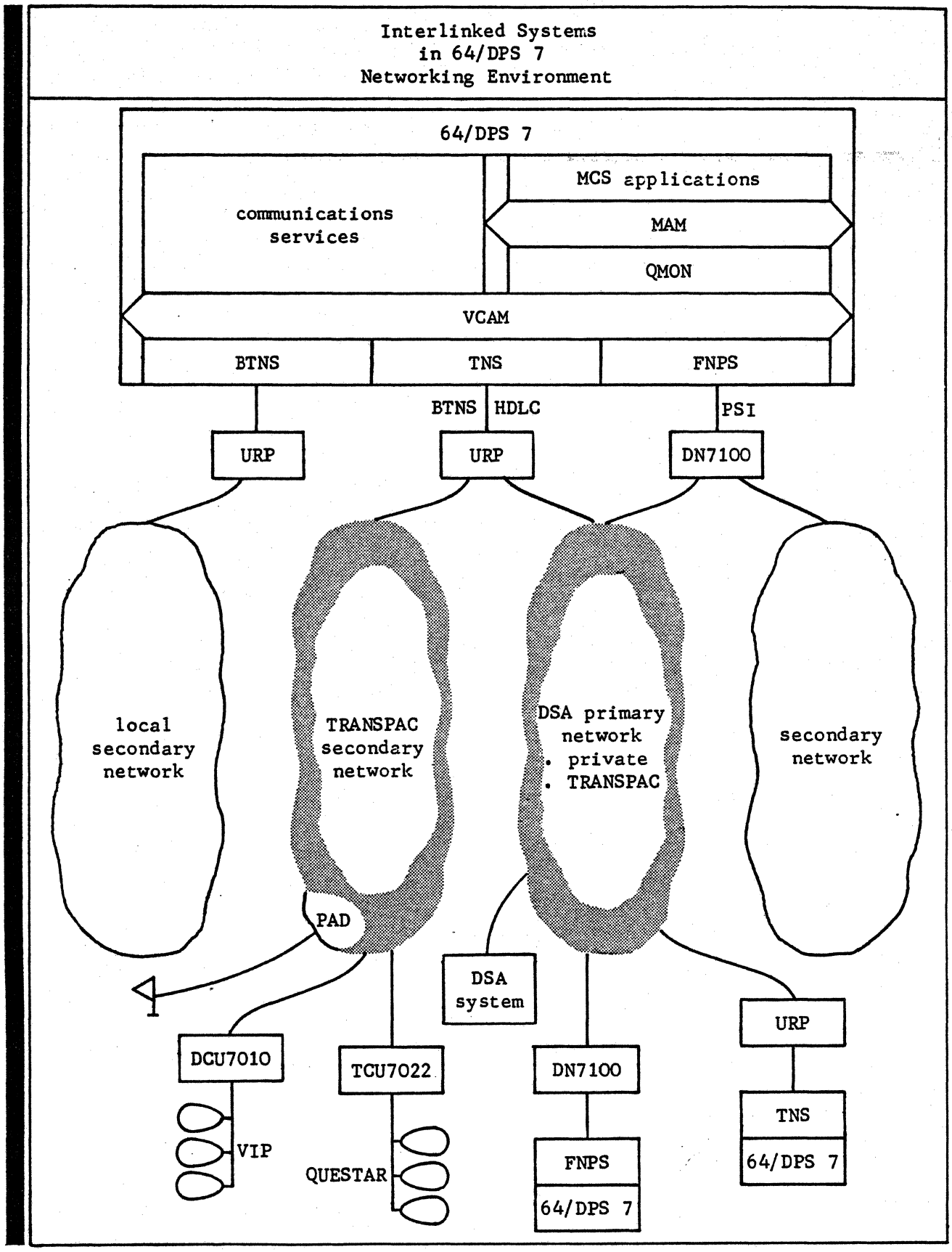
The DN7100 terminal manager supports both TWA and TWS session protocols, the latter allowing VIP KCT terminals, which include QUESTAR, to connect over the DN7100 to access MCS in the DPS 7 host.

Session control in both the DN7100 and the DPS 7 supports TWS protocol, thereby allowing the link-up of MCS applications residing in the DPS 7 host (local) and in its configured remote systems.

ACCESS TO MCS OVER BTNS/URP

For the secondary network, BTNS allows all standard terminals of the different line procedures to access MCS in the DPS 7. For as long as the line procedure is supported in the DPS 7, there are no connection restrictions.

For the primary network, the TNS function of BTNS/HDLC allows the link-up of MCS applications residing in the DPS 7 satellite (local) and in its configured remote systems.


Interlinked Systems
in 64/DPS 7
Networking Environment
64/DPS 7
communications services
MCS applications
MAM
QMON
VCAM
BTNS
TNS
FNPS
BTNS
HDLC
PSI
URP
URP
DN7100
local secondary network
TRANSPAC secondary network
DSA primary network
• private
• TRANSPAC
secondary network
PAD
DSA system
DCU7010
TCU7022
DN7100
URP
VIP
QUESTAR
FNPS
64/DPS 7
TNS
64/DPS 7

NETWORKING ENVIRONMENT

GCOS software allows remote users, such as terminal operators and applications located in other systems, to access

- user-defined applications which can be
  - MCS applications which are the subject of this manual
  - transaction programs, written in either COBOL or RPG, under TDS control
- GCOS communications services, otherwise known as VCAM subsystems, which are
  - RBF6 / FTF6, Remote Batch Facility / File Transfer Facility from / to the Mini 6
  - DJP / DFT, Distributed Job Processing / DSA File Transfer facility
  - IOF including the "pass-through" function operating under IOF
  - TDS, Transaction Driven Subsystem
  - CARDLESS, also known to the system as READER
  - TILS, Transactional and Interactive Load Simulator
  - OLTD, On-Line Tests and Diagnostics.

From a transmission standpoint, access to GCOS is through

- either the Data Communications Controller of the URP
- or the DN7100 functioning as a front-end processor.

In both cases, the URP and the DN7100 handle communications over secondary networks composed of local, leased and switched lines connected in a tree structure.

For the current release, the primary network can be configured with the DPS 7 functioning

- either as the "host" accessing the remote systems through the intermediary of its front-end processor, the DN7100
- or as the "satellite" accessing the remote systems directly through its URP.

The URP, in conjunction with TNS, a function of BTNS/HDLC, allows access to

- the TRANSPAC secondary network and the BTNS local network
- and, a DSA primary network linked either by virtual circuits over TRANSPAC or by point-to-point HDLC lines.

The DN7100, in conjunction with FNPS, can handle communications over

- secondary networks, like those supported over the URP
- and, such types of primary networks as, DSA high level networks and public networks, such as TRANSPAC.

The diagram opposite shows the extent of the communications interfaces in the DPS 7 networking environment.

For the network control operator's interface with the GCOS communications system, see the Network Control Terminal Operations Manual.

For the system and operations interfaces for terminals connected through the BTNS/URP secondary network, including TRANSPAC, see the Terminal Operations Manual.

COMMUNICATIONS COMPONENTS

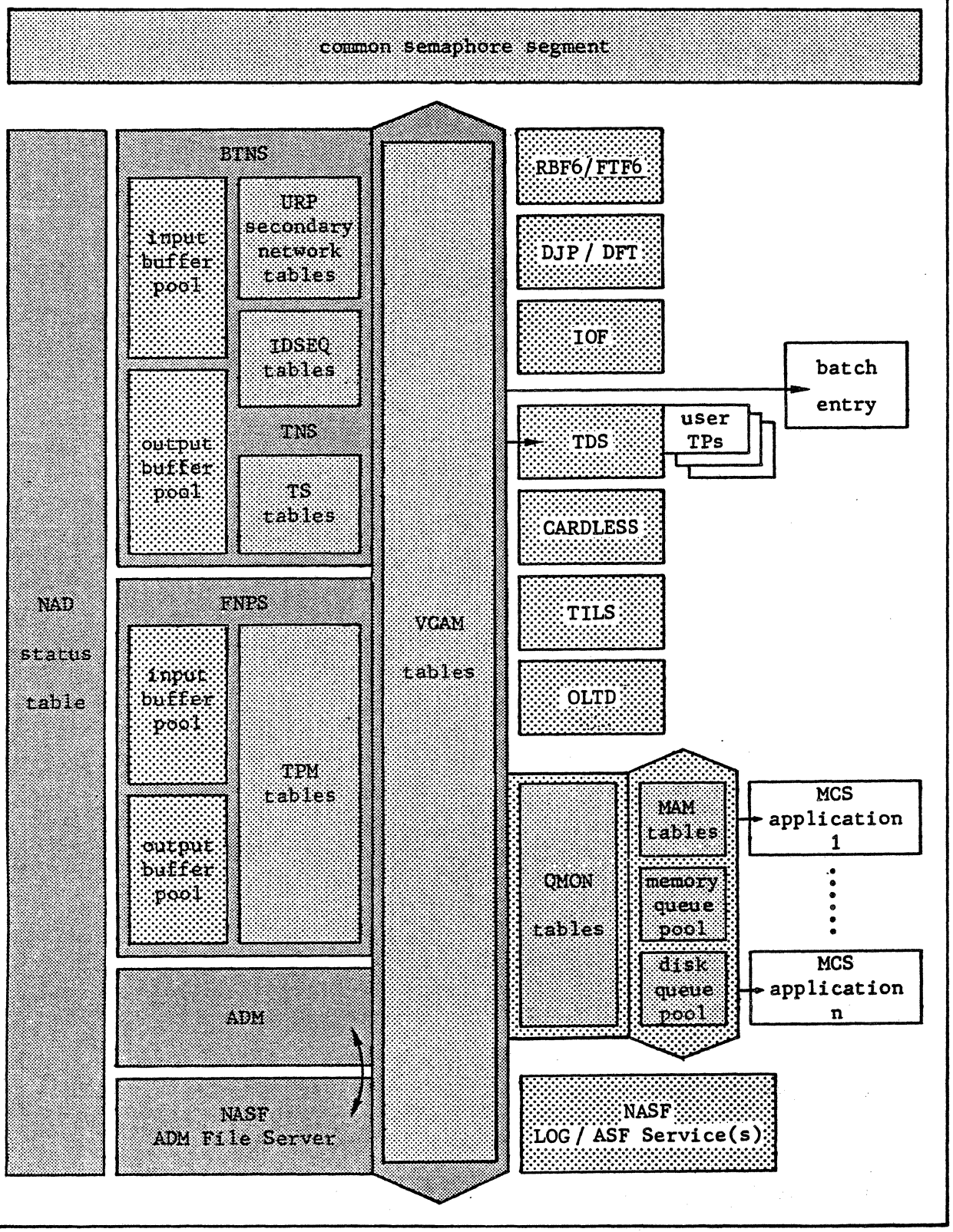
GCOS communications architecture is structured around the following three main DSA layers, namely

- the transport/network - link layer(s) occupied by communications management which comprises the following modules
  - BTNS for managing the terminals in the URP local network
  - TNS, a function of BTNS/HDLC
    - for managing the terminals in the TRANSPAC/URP secondary network accessed either directly over synchronous links or over PAD
    - and, for providing the direct interface over the DCC of the DPS 7 to the primary network thereby enabling the DPS 7 to function as a "satellite"
  - FNPS for interfacing at transport level with the DN7100 as front-end processor to the DPS 7 thereby enabling the DPS 7 to function as a "host" in the primary network
- the session layer occupied by VCAM for providing the interface between the communications management on the one side, and the application layer on the other side, such as
  - handling connection and dialog functions
  - and, allowing direct access to terminals and applications without their being aware of the communications path or mechanism used for establishing their connection
- the application layer occupied by
  - communications services, otherwise known as VCAM subsystems, which include MCS, being QMON operating with MAM
  - and, user applications which execute under either MCS or TDS, namely
    - MCS applications, which are the subject if this manual
    - TDS transaction programs
    - and, batch entries to TDS over the VCAM interface.

In "Schematic Interfaces of DPS 7 Communications Components",

- communications management and VCAM are shown attached together since they complement each other's functions to operate jointly as the network management component
- ADM/NASF (ADM file server) are mutually exclusive to each occurrence of the FNPS service corresponding to the DN7100 configured on the DPS 7 system, that is, for a given DN7100 either service functions are performed on it through ADM/NASF (ADM file server) or normal operations are executed on it when its associated FNPS service is started
- the NASF component providing LOG and ASF services functions as follows
  - the LOG service functions like any of the other communications services, such as TDS or MCS or IOF
  - the ASF service, however, functions as a communications process group which includes all components of communications management and QMON
- the NAD status table is a list of entries for each communications process group
- tables and pools which can be defined by the user through the CNC utility are shown where appropriate.

Schematic Interfaces
of DPS 7
Communications Components



MESSAGE CONTROL SYSTEM

MCS is a GCOS communications service whose functions are provided by

- MAM which interfaces user-defined applications written in MCS COBOL or GPL with MCS queues
- QMON which ensures the interface between the queues and VCAM, being the common unique interface between all GCOS communications services and remote users.

MCS allows the MCS application to communicate with

- terminals connected over secondary networks such as
  - the URP local and TRANSPAC/URP secondary networks
  - the FNPS/DN7100 secondary network
- other MCS applications located in either the same DPS 7 or in other DPS 7's in the following types of networks
  - either through the BSC2780 link over the BTNS/URP secondary network
  - or over the primary network accessed by the DPS 7 local system through either the TNS/URP interface or the FNPS/DN7100 interface, see page 1-01.

Details of connection handling are treated in Section IV.

Message Access Method

MAM provides such functions as,

- compatibility through MCS with standard communications language elements, namely, [$H_]CD, [$H_]ENABLE, [$H_]DISABLE, [$H_]SEND, [$H_]RECEIVE and ACCEPT/$H_MSGCNT verbs, shown in the form of MCS COBOL and GPL
- access to memory and disk queues, with 4 levels of queue available for input
- checkpoint/restart capability for disk queues
- allocating queues to the user application
- message editing according to the terminal type
- end-to-end protocol between the terminal operator and the user application, as provided either by control messages generated by MAM or by the communications status generated by BTNS
- communication between process groups, that is, communications load modules, and between processes, that is, tasks
- multitasking within the user application from 1 through 6 user processes.

A detailed description of MCS is given in Section II.

Queue Monitor

The QMON service workstation is the set of all user-mailboxes handling MCS queues, each queue having its own mailbox and bearing the same name.

The QMON service mailbox is known to GCOS by the system name QMONMBX.

QMON is in charge of establishing the logical connection

- between the application-mailbox, associated with the program-queue for input
- and the terminal-mailbox, associated with the terminal-queue for output.

Once the connection is established, data exchange takes place as follows,

- QMON receives the data from the terminal-queue and forwards it to the corresponding mailbox
- data forwarded to the application-mailbox is placed by QMON into the program queue.

In addition to connection handling, QMON performs other functions, such as,

- the transformation of data into and from mark form representation according to the transmission mode options specified for the terminal-queue
- automatic editing on messages sent to the terminal-mailbox according to the editing options specified for the terminal-queue, and which affect
  - message length, beyond which the message is truncated
  - blocking by line and by page.

Transmission mode and editing options specified for the terminal-queue at network generation through the QUEUE command can be overridden by the [$*$]MTE network control or terminal operator command during the communications session.

MCS Data Formats

MCS provides the means of representing data in a number of ways, which frees the user from the constraints of terminal hardware capability, for example, the user can specify lower-case letters through symbolic representation, although the terminal that he uses for data entry does not have the lower-case option.

In addition, the ability to encode control codes in mark form enables the MCS application to deal with the appropriate functions to be generated for a given terminal.

Data processing by MCS is treated in detail in Section III.

Programming Terminals

Terminals operate on a line procedure, which is used to establish transmission protocols over the link.

Transmission protocols are seen only at system level and do not affect the user.

User visibility in managing terminals of the network is limited to programming their control codes by which they function.

In Section VI, "Programming Terminals", the control codes are given in mark form instead of numeric values, for ease of mnemonic recognition and arrangement in alphabetical order.

QUEUE MAINTENANCE

QMAINT is a system utility used exclusively for MCS applications for executing maintenance actions on memory and disk queues.

It performs such functions as,

- printing the contents of the queue
- displaying the status of the queue
- purging the queue
- filling the queue with defined data.

For details of QMAINT, see Section VII and Appendix B.

SECTION II

MESSAGE CONTROL SYSTEM

The Message Control System is the interface

- between MCS applications and terminals accessed over BTNS and FNPS
- between MCS applications residing in the local and remote systems accessed either through FNPS or TNS, a function of BTNS/HDLC.

MCS functions are provided by MAM and QMON, as follows,

- MAM provides a user-queue access method shared at system level allowing the exchange of messages to and from the terminals or applications
- QMON ensures the interface between MCS queues and the basic communications functions provided by VCAM.

MCS applications can be written in either MCS COBOL or GPL.

MCS is described in terms of

- communications elements
- queue correspondence
- device handling and message editing
- dialog handling
- data integrity
- structure of a MCS load module
- communication between local applications
- communication between remote applications
- executing MCS applications.

COMMUNICATIONS ELEMENTS

The communications elements are

- the communications description area of the application
- the communications verbs
- the message delimiters.

Communications Description Area of the Application

The CD specifies the interface area between MCS and the user application.

These interface areas contain information about the queues, terminals and messages on input as well as on output.

The MCS application must have at least one CD area either for input or for output.

If the application requires messages to be sent and received, then at least two CD areas must be present, one for input and the other for output.

CD entries are defined as follows,

- in MCS COBOL, in the Communication Section
- in GPL, by the system primitive H_CD.

Communications Verbs

The 5 following communications verbs provide the user interface between MCS on the one hand and either the application or the terminals on the other.

The status of the MCS interface is denoted by key codes described in Appendix D.

MCS COBOL	GPL primitive	function
ACCEPT	H_MSGCNT	ascertains the number of messages in a symbolic queue identified by the symbolic queue in the input CD area of the application.
DISABLE	H_DISABLE	terminates the logical connection with specified sources or destinations for data transfers to and/or from the terminals.
ENABLE	H_ENABLE	establishes the logical connection with specified sources or destinations for data transfers to and/or from the terminals.
RECEIVE	H_RECEIVE	requests a message from a specified symbolic queue identified by the symbolic queue in the input CD area of the application.
SEND	H_SEND	directs a message to a specified symbolic queue identified by the symbolic destination in the output CD area of the application.

Message Delimiters

A message is delimited by the END KEY which has the following values

- "1" for ESI, "end-of-segment" indicator
- "2" for EMI, "end-of-message" indicator
- "3" for EGI, "end-of-group" indicator.

The message delimiter is coded in its mnemonic form as follows

- in MCS COBOL, by the statement SEND WITH { ESI | EMI | EGI }
- in GPL, by the primitive $H_SEND ENDCHAR= { ESI | EMI | EGI }

The following examples denote the way in which message delimiters are applied, namely,

- from the terminal to the application, in input mode
- from the application to the terminal, in output mode
- communication between applications.

FROM THE TERMINAL TO THE APPLICATION

° For a non-BSC terminal, the message is delimited by an END KEY value of "3" or EGI.

° For a BSC terminal,

- each block is terminated by the control code "ETB", "end-of-transmission-block", and is treated by the application as being the END KEY value of "2" or EMI
- the final block terminating the message text is delimited by the control code "ETX", "end-of-text", and is treated by the application as being the END KEY value of "3" or EGI
- where the message is of "zero" length, the END KEY value can be either "2" or "3".

  In this case, after issuing a RECEIVE (H_RECEIVE), the programmer should test the values of the following parameters before deciding if there is a message to be processed, namely,

  - STATUS KEY
  - TEXT LENGTH
  - END KEY.

FROM THE APPLICATION TO THE TERMINAL

° For a non-BSC terminal,

- message segments may be indicated within the message by an END KEY value of "1" or ESI, in which case, each indicator appears on the terminal as a "new-line" or "line-feed" sequence
- the message can be delimited by an END KEY value of "2" or EMI, or, "3" or EGI
- where the QUEUE is declared with the TWA option at network generation, the END KEY value of "3" or EGI is necessary to allow for dialog.

° For a BSC terminal,

- the message is transmitted in blocks, the maximum size of which is dependent on the type of the receiving terminal
- a SEND (H_SEND) with EMI results in the transmission of a data block terminated by the control code "ETB"
- a SEND with EGI results in the transmission of a data block terminated by the control code "ETX".

COMMUNICATION BETWEEN APPLICATIONS

The ESI, EMI and EGI delimiters are transmitted by MCS and converted into the appropriate END KEY values of "1", "2" and "3" respectively, in the destination CD area when a RECEIVE (H_RECEIVE) is issued.

QUEUE CORRESPONDENCE

A queue is a container in which messages are stored and from which messages can then be retrieved for later processing on a first-in-first-out basis.

Depending on application requirements, the queue can be specified either in main memory or on a disk file.

Queue correspondence involves

- the definition of the physical queue
- the definition of the symbolic queue
- relating the physical queue to the symbolic queue.

Physical Queues

Physical queues are defined by QUEUE commands at CNC generation, and are identified by external-queue-names.

The physical queue can be

- either a terminal-queue
- or a program-queue.

TERMINAL QUEUE

The terminal-queue is an output queue through which a message is sent to the terminal.

The term "terminal-queue" throughout this manual refers to one of the following,

- the name of the terminal declared in the TERMNL command for a local terminal
- the name of a DSA-terminal-queue of the format <system-name.mailbox-name> declared in the QUEUE command for an application connected either over the secondary network or to a remote system
- the name of a userid-queue declared in the QUEUE command for a terminal not declared with the AUTO option in the TERMNL command.

PROGRAM QUEUE

The program-queue is an input queue through which a message is sent to the MCS application.

The name of the program-queue is the name of the application specified during the log-on of the terminal.

Symbolic Queues

Symbolic queues are logical queues defined as follows,

- in MCS COBOL, in data-name-1 of the CD area (input and output)
- in GPL, by QUEUE_NAME in either H_CDIN or H_CDOUT.

The queue can be

- either input, for message reception
- or output, for message dispatch.

In the case of the symbolic input queue, further partitioning into up to 3 levels of subqueues can be done as follows,

- in MCS COBOL, by data-name-2, data-name-3 and data-name-4 of the input CD area
- in GPL, by SUBQUEUE_NAME, SUBQUEUE2_NAME and SUBQUEUE3_NAME in H_CDIN.

Relating Physical Queues to Symbolic Queues

The $QASSIGN statement defines the symbolic queue or subqueue and establishes its correspondence with the physical queue identified by its "external-queue-name" in the QUEUE command at network generation, this correspondence being unique.

The $QASSIGN statement also defines the processing mode for each type of queue,

- symbolic input queues
- symbolic output queues
- symbolic input/output queues.

SYMBOLIC INPUT QUEUES

A symbolic input queue must be defined for each program-queue from which messages are to be received by the application.

The IN parameter of the $QASSIGN statement serves

- to identify the symbolic queue as an input queue
- to allocate the program-queue to the current step until step termination.

No other MCS application may issue a RECEIVE (H_RECEIVE) to the program-queue thus allocated.

SYMBOLIC OUTPUT QUEUES

Each terminal to receive output has an associated terminal queue for which a symbolic output queue is defined.

The two ways of defining the symbolic output queue are

- explicitly, through the $QASSIGN statement
- implicitly, at terminal log-on.

Explicit Definition :

The OUT parameter of the $QASSIGN statement serves

- to explicitly identify the symbolic queue as an output queue
- to allocate the terminal queue to the current step until step termination.

The terminal is known to the MCS application exclusively by its explicitly defined symbolic output queue.

When a message is received from this terminal, the symbolic source field of the input CD (H_CDIN) is updated with the symbolic output queue-name by MCS.

No other MCS application may issue a SEND (H_SEND) to this terminal queue and the terminal cannot be connected to another MCS application.

Implicit Definition :

The symbolic output queue is defined implicitly through the log-on procedure of the destination terminal.

When a message is received from the terminal, the symbolic source field of the input CD (H_CDIN) is updated by MCS with the name of the associated terminal queue.

The symbolic source name will be used as the symbolic output queue to which messages are sent.

This method cannot be used if no message is sent from the terminal to the application unless the program-queue is defined with the BREAK option, in which case, the logon of the terminal is notified to the application with the status key 9D upon RECEIVE (H_RECEIVE).

SYMBOLIC INPUT/OUTPUT QUEUES

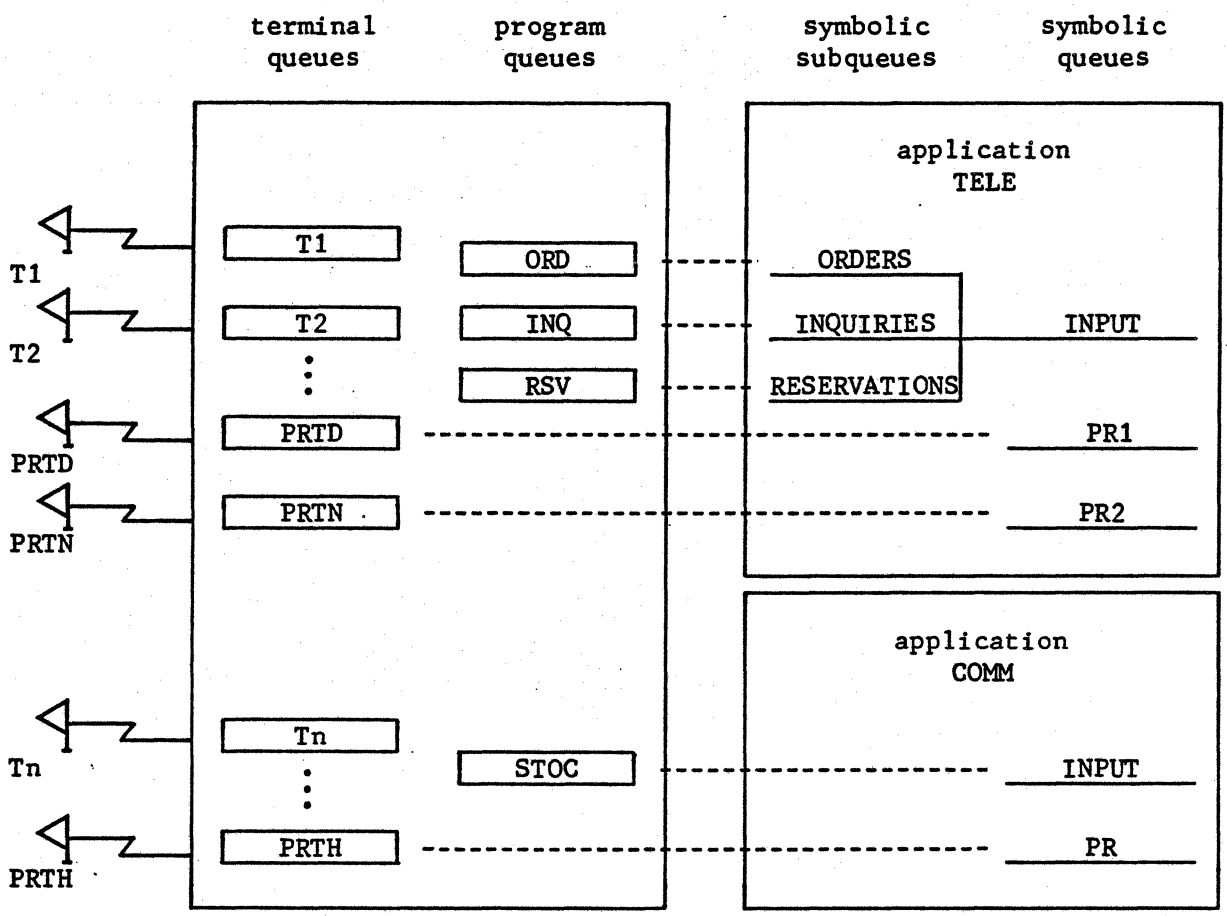
A symbolic I/O queue is a program queue and does not have subqueues.

The INOUT parameter of the $QASSIGN statement serves

- to identify the symbolic queue as an input/output queue
- to allocate the program queue to the current step until step termination.

No other MCS application may issue a SEND (H_SEND) or RECEIVE (H_RECEIVE) to the program queue thus allocated.

Example
of
Queue Correspondence



Application TELE queue relationship

- each of the program-queues is related to a corresponding symbolic subqueue, i.e., ORD to ORDERS, INQ to INQUIRIES, and RSV to RESERVATIONS.
- the symbolic subqueues ORDERS, INQUIRIES and RESERVATIONS are associated with the symbolic queue INPUT which means that when a RECEIVE (H_RECEIVE) is issued to INPUT, its subqueues are scanned until a message is found in 1 of them, e.g., INQUIRIES, if say, either T1 or T2 is logged on to INQ.
- the symbolic output queues PR1 and PR2 are related to their respective terminal-queues PRTD and PRTN associated with printers.

Application COMM queue relationship

- the direct correspondence of the program-queue STOC to the symbolic queue INPUT means that input is received from any terminal logged on to STOC
- the symbolic output queue PR is related to the terminal-queue PRTH associated with a printer.

DEVICE HANDLING AND MESSAGE EDITING

An MCS application is capable of such control functions as,

- sending control messages
- being notified of status changes
- controlling data flow.

Control Messages

Control messages are commands or status indicators affecting the MCS application-to-terminal interface, and can be passed between the application and QMON.

The application views the command like any other message that it sends to the terminal.

The format of the command is :

>\<CTLxxx, where xxx is a 3-character mnemonic variable for the command

The types of commands are,

- BRK : interrupt request
- CNT : indicates the terminal has been connected
- DIS : indicates the terminal has been disconnected
- PRG : application request to QMON to purge all existing messages in a terminal queue;
  this command is given top priority in the queue by QMON
- RVI : application request to issue a "reverse interrupt" to a BSC line procedure terminal related to the specified output queue;
  this command is stored in the queue and processed in sequence
- SHT : request to application to shutdown.

When a control message arrives in the destination queue, the count of available messages in the queue is incremented by 1.

The message type is detected by the "receiver" and reflected by a specific value of the STATUS KEY code.

On completion of a RECEIVE (H_RECEIVE), both parameters TEXT LENGTH and END KEY will be 0.

The particular use of the commands is as follows,

- BRK, CNT and DIS commands should only be used for terminal simulation purposes.
  These commands when sent to the terminal will be received by the terminal but will have no effect.
  The application, on receiving these commands, will be notified by the corresponding STATUS KEY code in the input CD area
- SHT command can also be sent by the BT network control command of the format : BT program-queue-name >\<CTLSHT.

Status Changes

If the program-queue has been defined with the BREAK option in the QUEUE command at CNC generation, status changes of the terminal or of the system will be passed to the application through the STATUS KEY of the input CD area as a result of a RECEIVE (H_RECEIVE).

The symbolic source identifies the related terminal.

If BREAK has not been specified, then the application will not be notified of any events occurring when a RECEIVE (H_RECEIVE) is issued.

STATUS KEY codes are explained in detail in Appendix D.

Data Flow Control

The control of data flow between MCS and the terminals is through the ENABLE (H_ENABLE) and DISABLE (H_DISABLE) verbs functioning as follows,

- input without terminal
- input with terminal
- output.

INPUT WITHOUT TERMINAL

° Enable Input :

It is coded as follows,

- in MCS COBOL, by the statement ENABLE INPUT
- in GPL, by the primitive $H_ENABLE INPUT.

The related program-queue specified in the input CD area is "enabled", i.e.,

- connection requests from the terminals can now be accepted
- terminals defined with AUTO and ASSIGNed to one of the queues or subqueues will be immediately connected when the RT network control command is issued, if required.

° Disable Input :

It is coded as follows,

- in MCS COBOL, by the statement DISABLE INPUT
- in GPL, by the primitive $H_DISABLE INPUT.

The related program-queue specified in the input CD area is "disabled", that is,

- no more connections to the queue can be accepted
- any terminals previously connected are now disconnected.

The application may continue to empty the queue through RECEIVEs (H_RECEIVEs) and when the queue is empty, the STATUS KEY is flagged as "disabled".

INPUT WITH TERMINAL

- Enable Input with Terminal :

  It is coded as follows,

  - in MCS COBOL, by the statement ENABLE INPUT TERMINAL
  - in GPL, by the primitive $H_ENABLE INPUT TERMINAL.

  The terminal whose name is specified as the symbolic source in the input CD area can start or resume input to the queue to which it was connected.

- Disable Input with Terminal :

  It is coded as follows,

  - in MCS COBOL, by the statement DISABLE INPUT TERMINAL
  - in GPL, by the primitive $H_DISABLE INPUT TERMINAL.

  The terminal whose name is specified as the symbolic source in the input CD area can no longer transmit in input mode.

OUTPUT

"Enable output" and "disable output" only apply to terminal queues.
If either verb is applied to program-queues, no action results.

- Enable Output :

  It is coded as follows,

  - in MCS COBOL, by the statement ENABLE OUTPUT
  - in GPL, by the primitive $H_ENABLE OUTPUT.

  The related terminal-queue specified in the output CD area is "enabled", that is, output flow will be resumed to the terminal whose name is specified in the output CD area.

- Disable Output :

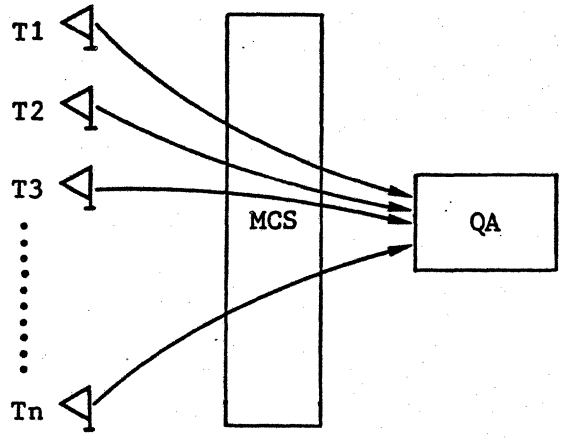
  It is coded as follows,

  - in MCS COBOL, by the statement DISABLE OUTPUT
  - in GPL, by the primitive $H_DISABLE OUTPUT.

  The related terminal-queue specified in the output CD area is "disabled", that is,

  - output flow from the queue to the terminal is suspended
  - the application may continue to send messages to the queue.

Examples
of
ENABLE

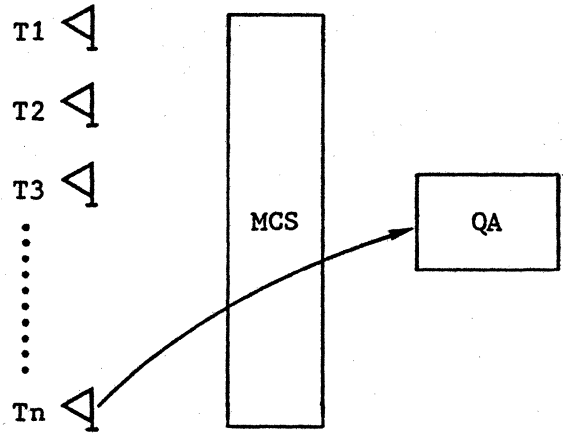


ENABLE INPUT

ENABLE INPUT with input CD area specifying the related program queue QA.

The queue QA is "enabled", whereby

- connection requests from terminals to QA and its subqueues are accepted
- terminals with AUTO and ASSIGN are immediately connected to QA and its subqueues when the RT network command is issued.

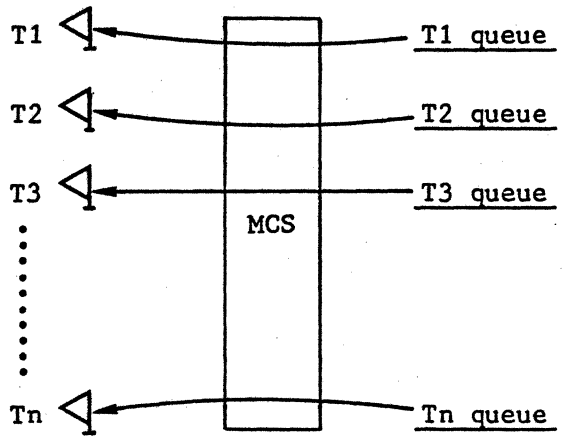


ENABLE INPUT TERMINAL

ENABLE INPUT TERMINAL with input CD area specifying the terminal Tn as the symbolic source.

The terminal whose name is specified as the symbolic source in the input CD area can start or resume input to the queue QA to which it was connected.

Other terminals continue to remain in their original states.



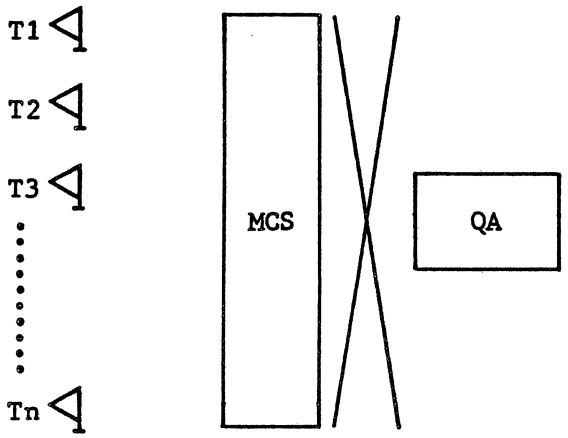
ENABLE OUTPUT

ENABLE OUTPUT with output CD specifying the terminal Tn as the destination of the output flow from its related queue.

Data flow can now resume from the queue specified in the output CD area to its related terminal.

Both the terminal and its related queue bear the same name, this correspondence being unique.

Examples
of
DISABLE

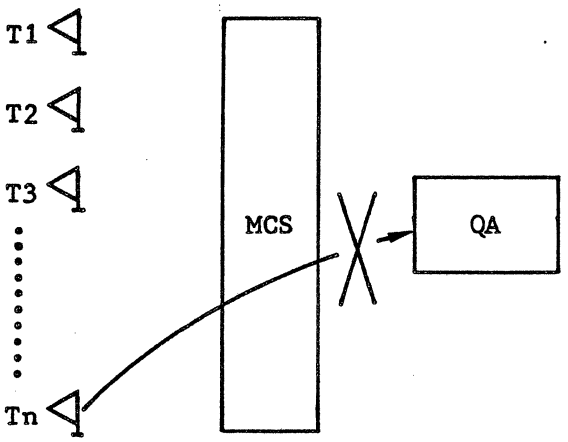


DISABLE INPUT

DISABLE INPUT with input CD area specifying the related program queue QA.

The queue QA is "disabled", whereby

- no more connections to the queue can be accepted
- any terminals previously connected are now disconnected.

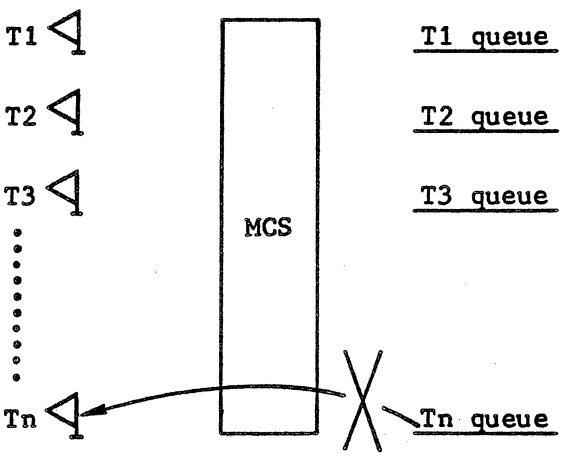


DISABLE INPUT TERMINAL

DISABLE INPUT TERMINAL with input CD area specifying the terminal Tn as the symbolic source.

The terminal whose name is specified as the symbolic source in the CD area can no longer transmit in input mode to the queue QA to which it is connected.

Other terminals continue to remain in their original states.



DISABLE OUTPUT

DISABLE OUTPUT with output CD specifying the terminal Tn as the destination of the output flow from its related queue.

The related queue Tn is "disabled"

- output flow from the queue to the terminal is suspended
- the application may continue to send messages to the queue.

DIALOG HANDLING

Dialog between the application and the terminal is handled according to the type of application, namely,

- non-interactive applications
- interactive applications.

Non-interactive Applications

A typical non-interactive application involves the following stages,

- data entry
- batch processing
- data distribution.

These stages are independent of each other, and the only relationship they bear to each other is that they are consecutive in the order shown.

The dialog between the terminals and their related queues, and between the application and the data entry queues is not restricted, that is, data flow is permitted in either direction.

DATA ENTRY

The application may be absent at the time when data is collected from the terminals into the program-queues.

Only the communications components, BTNS or FNPS, and QMON, need be present in system and the terminals need be connected to the program-queues, for data entry to take place.

In order to allow the terminal to enter data without the application being present, the program-queue must be defined as a data entry queue, that is, the TWA option must be omitted from the QUEUE command at CNC generation.

BATCH PROCESSING

When all the data has been entered, the application can begin processing during off-peak hours, say, late at night.

Processing involves updating files and preparing output data to be placed into the terminal queues.

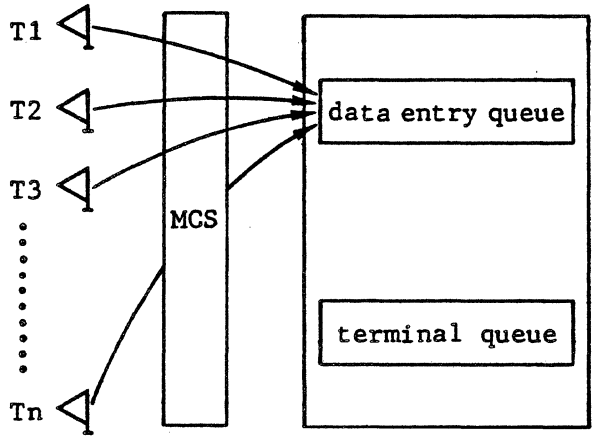
QMON plays no part in batch processing and is therefore absent from the system.

DATA DISTRIBUTION

QMON empties the output data in the terminal-queues to the respective terminals when these terminals are activated.

The application plays no part in data distribution and is therefore absent from the system.

Example
of
Non-interactive Application

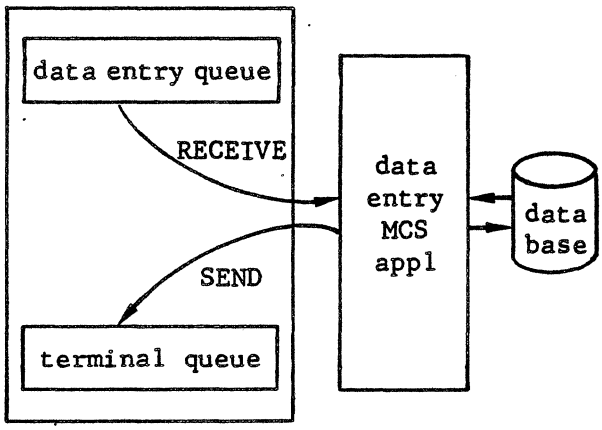


DATA ENTRY

All terminals can enter data at will to the data entry queue which is a program-queue declared without the TWA option.

Only BTNS or FNPS, and QMON need be present in the system.

The application does not need to be present in the system since the data entered is not immediately processed.

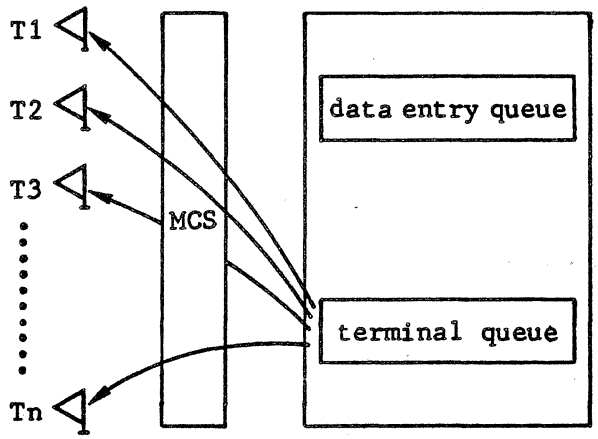


BATCH PROCESSING

Data collected from the terminals previously entered at the "data entry" stage can now be processed at a time when the terminals have ceased transmission.

Processing involves updating files and preparing output data to be put into the terminal queues for later dispatch.

QMON is absent from the system.



DATA DISTRIBUTION

When the terminals are activated, QMON outputs the results to the respective terminals.

The application plays no part in data distribution and can therefore be absent from the system.

The dialog between the terminals and their queues, and between the application and the data entry queues is not restricted.

Interactive Applications

Typical interactive applications are either transactional or inquiry-response applications.

The TWA option of the QUEUE command when declared at CNC generation for the associated queue determines the way in which the dialog is handled by the system.

An example of an MCS application testing the TWA option is given in Appendix A.

WITH TWA OPTION

The program-queue defined with the TWA option enables dialog between the application and the terminal on a message basis.

On connection, the terminal has the "turn", that is, the right to transmit.

A message transmitted by the terminal is delimited by the END KEY value of "3", denoting EGI.

The system, on detecting EGI in the terminal message, then transfers the "turn" to the application.

The application can then transmit using the SEND (H_SEND) verb.

Transmission by the application is performed as follows,

- several messages delimited by the END KEY value of "2", denoting EMI, can be transmitted and the "turn" is still retained by the application
- a message delimited by EGI transfers the "turn" to the terminal.

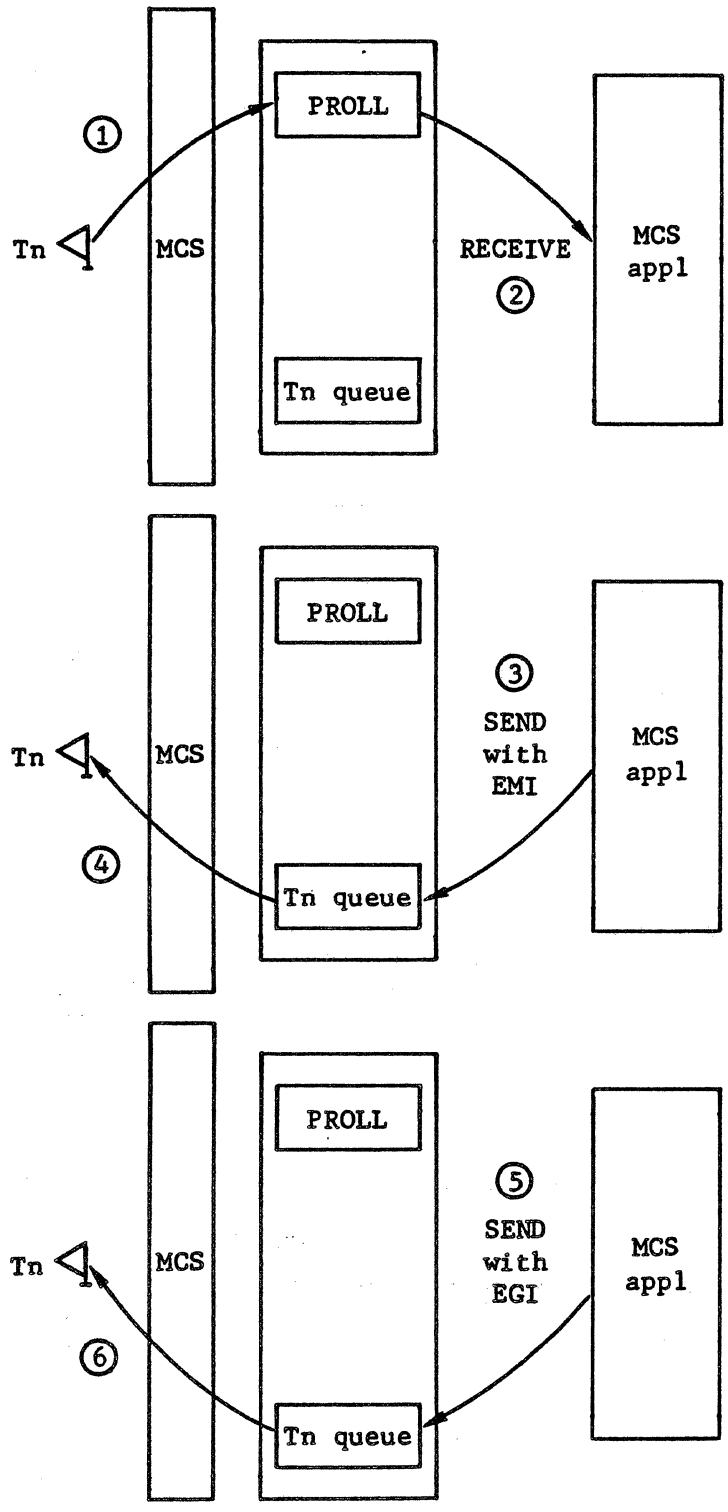
The terminal's "turn" can be overridden at any time by the application.

WITHOUT TWA OPTION

The program-queue defined without the TWA option enables the terminal to transmit messages at will.

In this case, the program-queue acts like a data entry queue.

Example of
Interactive Application
with TWA Option



1 Terminal Tn enters a message into the program-queue PROLL which has been declared with the TWA option at CNC generation.
The message is delimited by an END KEY value of "3" denoting EGI.

2 The message is RECEIVEd by the MCS application for processing.
In the meanwhile, the "turn" is given to the application.

3 After the MCS application has processed the message, it can SEND several messages delimited by EMIs to the terminal queue Tn.
For as long as messages from the application are delimited by EMIs, the application retains the "turn".
Messages from the terminal are discarded.

4 MCS starts transmission to the terminal Tn.

5 The last message in the stream to be sent by the application is delimited by EGI.
Although EGI transfers the "turn" to the terminal Tn, the application can still override the terminal.

6 When the terminal Tn receives the last message, it can then enter another message to the application.
If overridden, the terminal Tn can transmit later.

DECLARING AN APPLICATION "AUTOMATIC"

An automatic application is one which is automatically started

- if MCS finds that its associated queues contain data still to be processed
- and, if the application, for whatever reason, is no longer executing.

The facility for declaring an application "automatic" is indicated at network generation by both the following CNC options,

- the INIT parameter of the QUEUE command specifying the name of the application
- and, the APPLIB parameter of the GENQMON command specifying the cataloged library of application JCL subfiles, of which the referenced application is a member.

At run-time, if any queue of such an application declared "automatic" contains data for processing, and if the application has stopped executing, QMON

- identifies the application of the queue concerned from the argument of its INIT parameter
- retrieves the JCL subfile of the referenced application from the cataloged library specified as the argument of the APPLIB parameter
- and, starts the application.

The advantages of declaring an application "automatic" are that

- the application does not depend on the system console operator to be submitted
- and, even if the application were abnormally terminated either through a TJ system console command or as the result of a system failure, the application will be automatically resubmitted without further intervention of the operator.

The "automatic" facility is useful for an "interactive" application which must be present in the system for data to be entered and dispatched.

DATA INTEGRITY

The safeguard features in MCS to protect against the loss of data are,

- retention of messages in terminal-queues
- retention of messages in program-queues
- checkpoint and restart facility
- control of message and queue overflow.

Retention of Messages in Terminal Queues

When an MCS application issues a SEND (H_SEND), a STATUS KEY assigned to the activity is set to a value which can be tested by the user as follows,

- if the STATUS KEY denotes an incident, the message is not released
- if the STATUS KEY denotes normal conditions, then the message is released to the terminal queue under the charge of the system.

When the receiving terminal and its related communications path, over either BTNS or FNPS, are active, MCS attempts to transmit the message from the terminal queue to the corresponding terminal in the following way,

- if transmission is successful, the message is deleted from the terminal queue
- if transmission is not successful, several retries are attempted, according to the number specified by the user
- if transmission is persistently unsuccessful, the terminal queue is "closed" but the message is retained in the queue.

The message is retained in the queue until one of the following occurs,

- it is ultimately and successfully transmitted along the communications path to the terminal
- a shutdown is effected, whereby the message will be lost if the terminal queue is a memory queue or disk queue specified without the RESTART option
- a CNC step is executed, in which case, the message will be lost, even if the terminal-queue is specified with the RESTART option
- ISL with REFORMAT option is performed, in which case, the message will still be lost, even if the terminal-queue is specified with the RESTART option.

Message retention in terminal-queues applies under such conditions as,

- QMON "abort"
- system crash.

Retention of Messages in Program-Queues

A message which is successfully placed in a program-queue is retained until one of the following events occurs,

- it is successfully retrieved by the application for processing
- a shutdown is effected, whereby the message will be lost if the program-queue is a memory queue or a disk queue specified without the RESTART option
- a CNC step is executed, in which case, the message will be lost, even if the program-queue is specified with the RESTART option
- ISL with REFORMAT option is performed, in which case, the message will still be lost, even if the program-queue is specified with the RESTART option.

Checkpoint and Restart Facility

The facility allows the following functions,

- disk queues, which can be either program-queues or terminal-queues, specified with the RESTART option, are journalized and therefore can be "rolled back" to allow for recovery, as follows,
  - the head-of-queue of a program-queue is "rolled back" to the last valid checkpoint
  - the head-of-queue of a terminal-queue is "rolled back" to the last un-transmitted message
- disk terminal-queues specified with the CTLRST option allow for controlled "restart" for retransmission
- the user application specified with the REPEAT option in the $STEP statement can be restarted.

The conditions for which the facility is used are,

- step abort
- system crash.

The purpose of the facility is to ensure that the queues, the application, and all user files are in the same state when restart takes place, as they were at the last checkpoint.

If checkpoints have not taken place, then restart is at the beginning of the step.

The checkpoint and restart facility is dealt with under the following topics,

- issuing checkpoints
- implementing checkpoints
- conditions for restart.

ISSUING CHECKPOINTS

Only MCS monoprocess application with disk queues can issue checkpoints, as follows,

- in MCS COBOL, by the declarations,
  - RERUN ON CHECKPOINT FILE entry in the I-O-CONTROL paragraph, through which checkpoints are taken at specified intervals of records processed for a particular file
  - CALL statements to the H_CK_UCHKPT run-time package, by which checkpoints are taken at appropriate places in the application
- in GPL, by the primitives,
  - $H_FD specifying the CKPTLIM keyword, whereby checkpoints are taken at specified intervals of records processed for a particular file
  - $H_CHKPT which allows checkpoints to be taken at appropriate places in the application.

The program-queue is journalized if it has been specified as follows,

- with the RESTART option in the corresponding QUEUE command at CNC generation
- as an input queue, by either the IN or INOUT parameters of the corresponding $QASSIGN statement.

Journalization is performed on the disk queue itself and no additional file space needs to be allocated.

A journal is considered "active" for a given program-queue only for the duration of the MCS application step.

Checkpoints have no effect on terminal-queues except for terminal-queues corresponding to an HL64 CPU, to be discussed later on in "Communication between Remote Applications", see page 2-31.

IMPLEMENTING CHECKPOINTS

For program-queues described with the RESTART option, disk space related to messages will be released only at checkpoint time or on termination of the MAM application.

The user must take this mechanism into account,

- when defining the size of such program-queues in the corresponding QUEUE commands at CNC generation
- when deciding the frequency of checkpoints in his application.

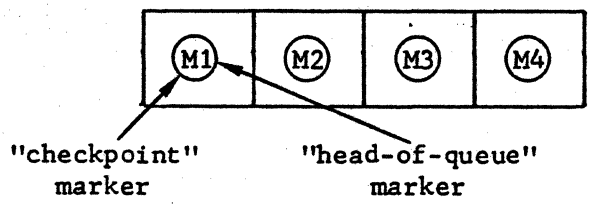
When reading messages from a program-queue, the "head-of-queue" marker points to the next message to be RECEIVEd by the application.

The "checkpoint" marker points to the position of the "head-of-queue" marker at the time of the last checkpoint.

In the case of application "restart", the "head-of-queue" marker is then restored to the "checkpoint" value.

Queue Status after Taking Checkpoints

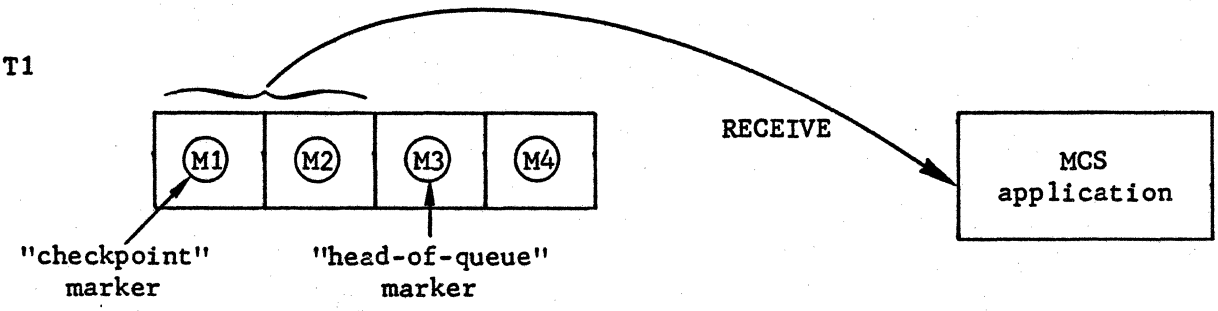
T0



At time T0, both "checkpoint" and "head-of-queue" markers point to message M1.

It is assumed that the queue holds 4 messages, and that at the start of the application, it is occupied by messages M1, M2, M3 and M4.

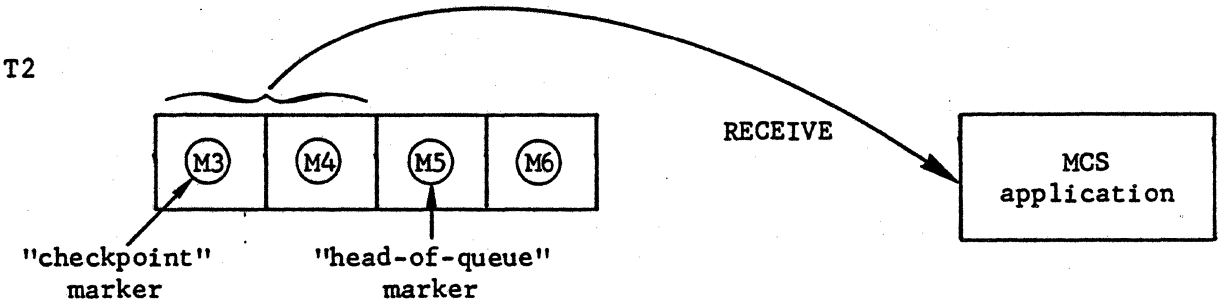
T1



At time T1, the situation concerning the queue status is,

- the messages RECEIVEd by the MCS application are M1 and M2
- the "head-of-queue" marker has moved from message M1 to point to message M3
- the "checkpoint" marker has not moved and still points to message M1.

T2



At time T2, when the MCS application has successfully processed messages M1 and M2, and taken a checkpoint, the situation concerning the queue status is,

- the "checkpoint" marker has moved from message M1 to point to message M3
- messages M5 and M6 now arrive in the queue displacing messages M1 and M2
- the MCS application has now RECEIVEd messages M3 and M4
- the "head-of-queue" marker has moved from message M3 to point to message M5.

If an abort occurs, the queue status at restart will be,

- the "head-of-queue" marker will be "rolled back" to the position of the "checkpoint" marker pointing to message M3
- the "checkpoint" marker, however, still points to message M3
- the messages to be RECEIVEd by the MCS application will be M3 through M6.

CONDITIONS FOR RESTART

A job step can only be restarted if the $STEP statement contains the REPEAT parameter.

In this case, the job step can be restarted either at the beginning of the step or from the latest checkpoint taken.

° Restart after Step Abort :

Queues assigned to the step with the RESTART option, are "rolled back" to the previous checkpoint if the step is restarted.

Otherwise, they are left in the state they were at the time of the abort, in which case, they can be printed out by the use of the QMAINT utility for debugging purposes.

All other queues are left unchanged.

° Restart after System Crash :

Queues assigned to the step are treated as follows,

- terminal and program-queues specified without the RESTART option are re-initialized
- terminal-queues defined with the RESTART option are restarted from the next message following the last message successfully transmitted.

  If the crash occurred during transmission, the queue is restarted with the message that was being transmitted at the time.

  Restart of terminal-queues with the CTLRST option, is dealt with in "Communication between Remote Applications", see page 2-31
- program-queues with an active journal are "rolled back" to the last checkpoint taken or to the beginning of the step
- program-queues with the RESTART option but without an active journal, that is, queues not assigned to the MCS application at the time of the system crash, are restarted from their current state.

The programming techniques to consider for restart after a system crash are,

- the MCS application restarted from the latest checkpoint will process the first message in the symbolic queue.

  This message may have been already processed and caused the delivery of an output message.

  In this case, the output message will be duplicated.
- if the application wants a terminal to be aware of its checkpoint, it can now send a message to the terminal indicating the checkpoint number.

  On restart from checkpoint "i", the application should send a message to the terminal indicating restart from checkpoint "i", so that the terminal operator is aware of the repetition of messages.

° Restart after System Crash (continued)

- messages being transmitted at the time of the crash should be retransmitted by the terminal operator at restart.

  The application should check for redundant messages by ensuring that all input messages should include sequence numbers.

- if the program-queue needs to be purged following a restart, the application may do so by issuing as many RECEIVEs as the message count.

° Restart at MAM initialization :

The choice of action for the system console operator when starting up MAM in the case where disk queues existed for the previous session are,

- MAM = YES
  - all queues without the RESTART option are purged
  - terminal-queues with the RESTART option commence following the message successfully transmitted or at the message being transmitted
  - program-queues with the RESTART option commence from
    - either the "head-of-queue" marker for normal termination of the session
    - or the last "checkpoint" marker, if the following conditions are fulfilled,
      - the queues have an "active" journal
      - the session terminated abnormally.

- MAM = NO

  This action is taken in cases where MAM is not required to save start-up time and space in system resident memory.
  - unless a new CNC session is run to generate the network, MAM is not available
  - the contents of the disk queues remain in the state they were at the time when the session terminated.

- MAM = REFORMAT
  - MAM start-up operations are performed whereby all queues are reinitialized
  - the disk queue file is reformatted using the copy of the communications system tables in backing store.

The programming techniques to consider for restart at MAM initialization are,

- program-queues can be filled during a session when no MCS applications are active
- the contents of such queues can then be retrieved in subsequent sessions when MCS applications are executed.

Control of Message and Queue Overflow

When the message size limit or queue capacity is exceeded, an overflow condition results which causes the STATUS KEY of the CD entry to be updated with the appropriate return code to inform the user.

The list of STATUS KEY codes is given in Appendix D.

CD entries are defined as follows,

- in MCS COBOL, in the Communication Section
- in GPL, by the system primitive H_CD.

MESSAGE OVERFLOW

The maximum message length is restricted to 3053 bytes.

Although the default value for QDBLKSZ is 400 bytes, it is advisable to declare at least 500 bytes when preallocating the disk queue file.

Specifying a QDBLKSZ value of less than 400 bytes results in a warning at CNC execution time and the value is then overridden by the default value of 400 bytes.

See "Preallocating a Disk Queue File" on page 2-43.

When message length, including control codes as hexadecimal values or in mark form, is exceeded, the excess portion of the message is truncated to 2432 bytes.

The STATUS KEY code setting is as follows,

- 94, MSGOV "message overflow" for the sender
- 94, NOTALL for the receiver.

The maximum message size should be defined by the application according to,

- the size of the terminal display
- the allowance to be made for transmission errors.

QUEUE OVERFLOW

When there is insufficient space in a queue to contain a message sent to it or to handle I/O transfers, the message is discarded and the STATUS KEY is set to the appropriate value.

When a queue overflow condition occurs on a SEND (H_SEND), the application should issue a "call" to the timer before attempting to re-execute that SEND.

The "call" to the timer is performed as follows,

- in MCS COBOL, by the statement CALL to the run-time package H_TM_USETTM
- in GPL, by the system primitive H_SETELT.

STATUS KEY code setting for the relevant condition indicating the type of queue overflow is,

- 91, SPACENAV, space not available, for the sender to be notified of the lack of space in the disk queue
- 92, BUFNAV, buffer not available, for the sender and the receiver to be notified that there is insufficient memory space to handle a disk I/O operation during a message transfer
- 95, NOTALL for the sender to be notified that the number of memory blocks is insufficient.

STRUCTURE OF AN MCS LOAD-MODULE

An MCS load-module comprises 1 through 6 user processes which are basically equivalent to tasks and are linked together by the Static Linker, see "Executing MCS Applications".

Each process is associated with an application or "run-unit".

The system handling of the MCS load-module depends on whether the module is monoprocess or multiprocess.

Monoprocess Load-Module

The system executes a call to the entry point of the monoprocess load-module specified by the user.

On termination of the process, a call is made to MCS to perform housekeeping functions, whether the process terminates normally or abnormally.

The program is terminated as follows,

- in MCS COBOL, by the EXIT PROGRAM statement
- in GPL, by the RETURN statement.

Multiprocess Load-Module

The system procedure STUSERS starts each of the STUSERn processes, where n ranges from 1 through 6.

Each STUSERn executes a call to the application through a user-specified entry point at linkage time.

On termination of the process, STUSERn calls MCS to perform housekeeping functions before returning control to STUSERS.

When all the processes have terminated, and all housekeeping functions have been performed, STUSERS terminates automatically.

The constraints for the multiprocess load-module are,

- the programmer must synchronize accesses to shared files, see "Communication between Local Applications"
- the checkpoint and restart facility is not available
- only one process can execute the ACCEPT (H_GET) and DISPLAY (H_PUT) verbs to gain access to the standard SYSIN and SYSOUT files.

In GPL, the user has the option of either using STUSERS as a system procedure or coding his own procedure whereby he can explicitly declare the start of each process.

This facility enables the GPL user to initiate secondary tasks not containing MCS primitives.

This topic is treated under "Executing MCS Applications", see page 2-43.

COMMUNICATION BETWEEN LOCAL APPLICATIONS

The term "local" means that the applications are in the same central processor, whereby communication is established by manipulating program-queues.

The program-queue whose name is to appear in the symbolic source of the destination input CD area is specified by the keyword REPLY in $QASSIGN OUT.

The ENABLE (H_ENABLE) and DISABLE (H_DISABLE) verbs are not effective.

The 3 types of local communication are,

- application-to-application communication
- application communicating with itself
- communication between processes of a multiprocess application.

Application-to-Application Communication

The assignments in the $QASSIGN statements required to allow such exchanges are,

Application A

- the symbolic queue INPUT must correspond to the program-queue QA for input, that is, specified with IN
- the symbolic queue OUTPUT must correspond to the program-queue QB for output, that is, specified with OUT and REPLY=QA

Application B

- the symbolic queue INPUT must correspond to the program-queue QB for input, that is, specified with IN
- the symbolic queue OUTPUT must correspond to the program-queue QA for output, that is, specified with OUT and REPLY=QB.

Application Communicating with Itself

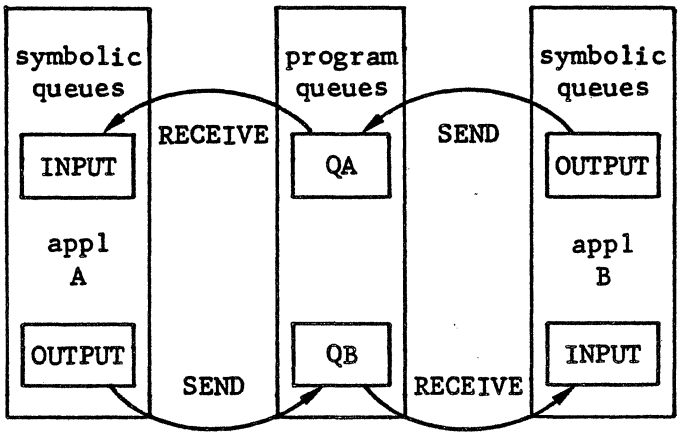
The assignment in the $QASSIGN statement required to allow such an exchange is,

- the symbolic queue INPUT to correspond to the program-queue QA for input as well as output, that is, to be specified with INOUT and REPLY=QA.

The application will then communicate with itself in the following stages,

- it will act as a terminal to fill the program-queue QA
- it will then RECEIVE messages from the same program-queue QA exactly as it would RECEIVE messages from any terminal connected to it.

Examples of Communication between Local Applications

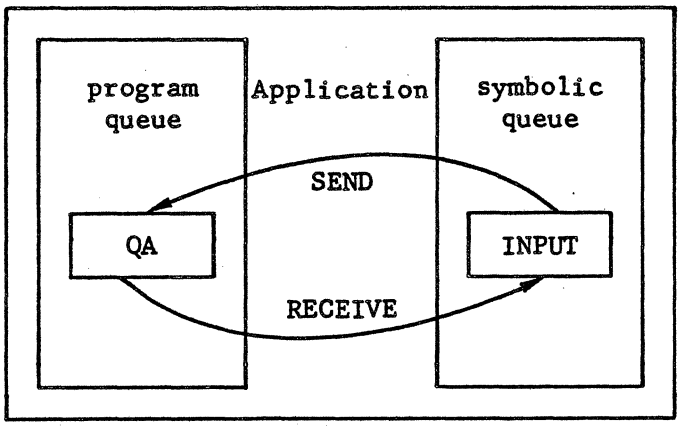


Application-to-Application Communication

Application A receives as input messages supplied by B, and vice versa.

For Application A, the symbolic queue OUTPUT must be specified with REPLY=QA.

For Application B, the symbolic queue OUTPUT must be specified with REPLY=QB.

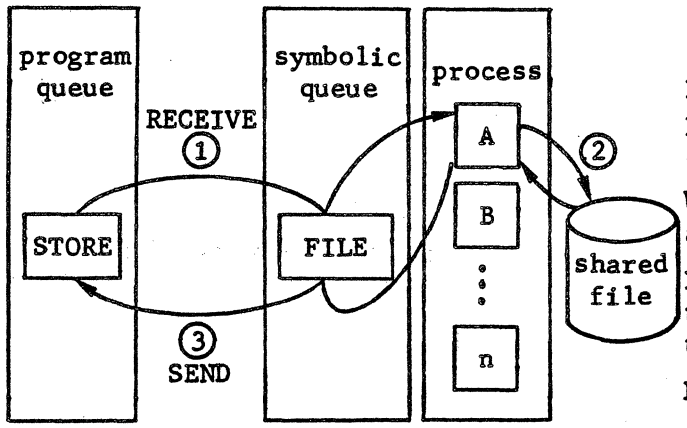


Application Communicating with Itself

Application acts as a terminal to fill the program-queue QA.

It then receives the messages from QA in the same way as it receives messages from any terminal connected to it.

The symbolic queue INPUT must be specified with REPLY=QA to allow for such an exchange.



Communication between Processes of a Multiprocess Application

Process A issues a RECEIVE to program-queue STORE through the symbolic queue FILE.

When acknowledged, Process A can access the "shared" file.

Process A then SENDs a message to STORE to notify Process B that the "shared" file is free.

Process B can access the file.

Communication between Processes of a Multiprocess Application

Such communication involves essentially the sharing of a file or files between processes of the same application.

The assignment in the $QASSIGN statement required to regulate exchanges between the processes accessing a "shared" file is,

- the symbolic queue FILE to correspond to the program-queue STORE for input as well as output, that is, to be specified with INOUT.

The programmer is responsible for synchronizing all accesses to the "shared" file for multiprocess applications.

The same mechanism can also be applied to a queue providing access to several applications belonging to different steps by specifying the SHARE option in the relevant QUEUE command at CNC generation.

The stages in implementing file sharing are,

- at step execution, the application must first prime the program-queue STORE with a message which serves to notify the first process wishing to access the "shared" file that is available
- the first process, process A, say, can then issue a RECEIVE (H_RECEIVE) to the program-queue STORE through the symbolic queue FILE.

  When the message is RECEIVEd, process A has access to the "shared" file.
- when process A has finished with the "shared" file, it SENDs a message to the program-queue STORE to notify any other process, process B, say, wishing to access the "shared" file that it is now available.

COMMUNICATION BETWEEN REMOTE APPLICATIONS

The term "remote" means that the applications are in different central processors which are either linked through a BSC line procedure over the BTNS/URP interface in the secondary network, or connected in the primary network over either the TNS/URP interface or the FNPS/DN7100 interface through the HDLC X.25 link.

The implementation of the BSC2780 line procedure is treated in detail in Section V "Line Procedures".

Information on the TNS/URP and FNPS/DN7100 interfaces is given in detail in Section IV "Connection Handling".

The following text is a description of how the BSC2780 link is handled.

Communication is established at any given time by only two applications at either site by means of describing the other site as a BSC terminal with the following attributes in the respective TERMNL command at CNC generation,

- terminal-type being CPU
- AUTO
- ASSIGN=program-queue, for which a QUEUE command bearing the name of the "program-queue" must be declared in the same CNC generation.

Communication between remote applications is described in terms of

- implementation
- recovery on the emission side
- recovery on the reception side

Implementation

The operation of the BSC link-up depends on such factors as,

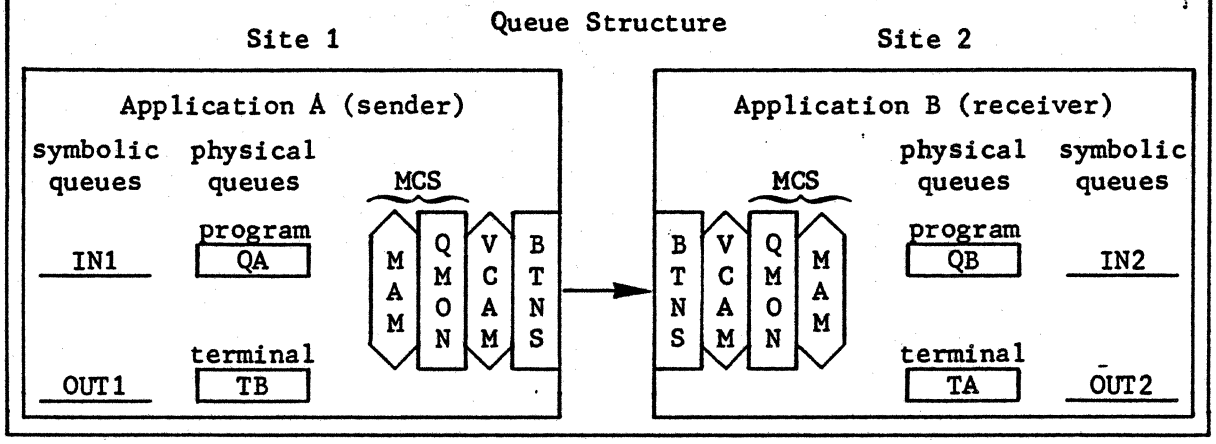
- the nature of the communication, that is,
  - either interactive communication
  - or unidirectional communication
- the recovery protocol.

The implementation of the link-up is achieved by

- the CNC description of the network
- the run-time JCL
- the message flow between the sender and receiver applications.

Setting up Communication between Remote Applications

Queue Structure



CNC Description for BSC Link-up

Site 1

```
LINE  LNnn  CLOSE;
STATN  STA1  1;
TERMNL  TB  HL64  CPU  ASSIGN=QA
        AUTO;
QUEUE  QA  ...  RESTART;
QUEUE  TB  ...  CTLRST;
```

Site 2

```
LINE  LNnn  CLOSE;
STATN  STA2  2;
TERMNL  TA  HL64  CPU  ASSIGN=QB
        AUTO;
QUEUE  QB  ...  RESTART;
QUEUE  TA  ...  CTLRST;
```

Run-time JCL

Site 1

```
QASSIGN  IN1  QA  IN;

QASSIGN  OUT1  TB  OUT;
```

Site 2

```
QASSIGN  IN2  QB  IN;

QASSIGN  OUT2  TA  OUT;
```

NATURE OF COMMUNICATION

° Interactive Communication :

Exchanges of data will take place in both directions successively from the sender to the receiver, and vice versa, the applications reversing roles.

However, before the applications can be started up, the link-up must first be established.

° Unidirectional Communication :

As in the case of file transfer, the sender application can be started up first, provided that

- its output queue is on disk
- and that the disk has enough space to contain the entire file to be transferred.

In some cases, the rate at which the data is sent exceeds the rate at which it is received.

This condition happens when the rate at which the output queue is being filled in by the sender application is greater than the rate at which the same output queue is being emptied to the receiver application.

When this occurs, as in the case of file transfer from magnetic tape, the sender application will encounter "queue-overflow" incidents.

To avoid such incidents, the sender application should issue a call to a system procedure to temporarily suspend transmission of data for a specified lapse of time before attempting to issue another SEND to the output queue.

In the meanwhile, transfers of data from the output queue to the receiver application can proceed unaffected.

A "loop" on the SEND (H_SEND) has the effect of a temporary suspension, although this method is not advised as it takes up too much CPU time.

RECOVERY PROTOCCL

Where the link-up between two remote applications involves a one-way transfer, recovery protocol uses the system checkpoint and restart facility.

The sender application emits a recovery mark each time it takes a checkpoint.

When the receiver application receives the recovery mark in the form of a control message, it, in turn, takes a checkpoint.

Restart is always initiated by the sender application.

In order to implement the restart facility, the following options must be declared in the appropriate QUEUE commands at CNC generation,

- program-queues must be defined with the RESTART option
- terminal-queues must be defined with the CTLRST option.

Communication between Remote Applications
Programming Message Flow
between Sender and Receiver

Site 1 Application A (sender) ⟶	Site 2 Application B (receiver)
① START : Call checkpoint. Test value of MODE. [$H_]SEND ("><CTLCKP00") EMI before SENDing data, A takes first checkpoint to SEND control message showing checkpoint numbered "00" (zero).	
MODE = 00	
② [$H_]SEND (data) {EMI | EGI}	
A SENDs 1 or more files terminated by the appropriate indicator.	
	ⓐ [$H_]RECEIVE Call checkpoint. first message RECEIVEd is checkpoint "00"; B takes its own checkpoint; both checkpoints have initial values "00".
	ⓑ [$H_]RECEIVE B RECEIVEs data from A as a stream.
③ Call checkpoint. Test value of MODE. [$H_]SEND ("><CTLCKPnn") EMI A takes sequentially numbered checkpoints to SEND as control messages.	
MODE = 00	
④ [$H_]SEND (data) {EMI | EGI}	
A continues to SEND data in a stream.	
	ⓒ [$H_]RECEIVE Call checkpoint. When B RECEIVEs successive checkpoints, it takes its own checkpoints. The result is that the sets of checkpoints of A and B match 1-for-1.
	ⓓ [$H_]RECEIVE B continues to RECEIVE data.
⑤ Call checkpoint. Test value of MODE [$H_]SEND ("><CTLCKPnn") EGI A takes final checkpoint and SENDs the control message terminated by EGI.	
MODE = 00	
⑥ End of normal transmission.	
	ⓔ [$H_]RECEIVE Call checkpoint. When B RECEIVEs EGI, it takes its final checkpoint.

Recovery on the Emission Side
Example of
Restart of Application on Incident

Site 1 — Application A (sender) ⟶ Site 2 — Application B (receiver)

Transmission has so far been normal and free of errors, but has not yet terminated.

① [$H_]SEND (data) {EMI|EGI}

A continues to SEND data in a stream.

② Call checkpoint.
Test value of MODE.
$H_ SEND ("><CTLCKP(ii)") EMI

MODE = 00

③ [$H_]SEND (data) {EMI|EGI}

step abort or system crash occurs

④ Test value of MODE.

MODE ≠ 00

[$H_]SEND ("><CTLRST(ii)") EMI

The number of the "restart" control is the same as the last valid checkpoint.

⑤ [$H_]SEND (data) {EMI|EGI}

ⓐ [$H_]RECEIVE

B continues to RECEIVE data.

ⓑ [$H_]RECEIVE
Call checkpoint.

The number of the checkpoint taken by B is also "ii".

ⓒ [$H_]RECEIVE

At the time of step abort or system crash, the data RECEIVEd by B is in an indeterminate state.

ⓓ [$H_]RECEIVE

When B RECEIVEs the "restart" control, it sets the STATUS register to 10,000 to denote "step abort" or "system crash".

EXIT PROGRAM : MCS COBOL
RETURN : GPL

ⓔ [$H_]RECEIVE

Recovery on the Emission Side

Recovery of transmission on the emission side involves,

- taking checkpoints
- sending control messages
- restarting after a step abort or system crash
- restarting after a line failure.

TAKING CHECKPOINTS

Each transmission is started with a checkpoint call.

On return of the call, the output parameter MODE is tested as follows,

- MODE = 00, for normal error free transmission
- MODE ≠ 00, when a step abort or system crash has occurred.

Taking checkpoints is performed as follows,

- in MCS COBOL, by a call to the H_CK_UCHKPT run-time package
- in GPL, by the system primitive H_CHKPT.

SENDING CONTROL MESSAGES

Control messages of the format "><CTLCKPnn" are used to head data flow.

During a session, one or several files can be transmitted, each file being terminated with an EGI.

To allow for recovery, the sender application takes checkpoints at suitable and regular intervals, and checks that the value of MODE is zero in order to continue with normal transmission.

The checkpoint is then sent as a control message to head the next stream of data flow.

When the checkpoint and the accompanying data stream are completely received with out error, the system will release the disk space related to all the messages sent since the last valid checkpoint.

The maximum number of disk blocks which can be retained between 2 checkpoints on emission is given by the algorithm,

$$n = \frac{(QDBLKSZ - 4)}{2}$$

where QDBLKSZ is the size of the disk block in bytes, declared in the GENCOM command at CNC generation

SENDING CONTROL MESSAGES (continued)

For transparent mode, the sender application must only take a checkpoint at the start of transmitting a file and not during transmission.

In this case, "><U06" must head the stream of data flow, see BSC2780 line procedure, page 5-13.

RESTARTING AFTER A STEP ABORT OR SYSTEM CRASH

The "restart" control message can be emitted from one of two sources,

- either from the sender application in the case of a step abort
- or from the system, in the case of a system crash for remote communications.

The "restart" control message serves to warn the receiver application that all messages sent since the last valid checkpoint will be re-transmitted.

The receiver application must therefore be programmed to avoid the duplication of messages.

If at the time of system restart, the sender application is also to be restarted, then the receiver application will see two restart phases in the following sequence,

- firstly, it will receive the "restart" control message from the system
- then, it will receive the "restart" control message from the sender application.

RESTARTING AFTER A LINE FAILURE

A line is closed by the system if failure occurs

- either due to a malfunction in the link-up
- or at the receiver site, for whatever reason.

When the failure has been rectified, the line can be re-opened by the network control command RT LNnn, where nn specifies the line.

The effects of this command on a BSC line are,

- the line will be reactivated
- the control message "><CTLRSTnn" (ETB) will be sent
- the terminal queue will be restarted from the last valid checkpoint indicated by nn of the "restart" control message.

Transmission will resume on successful emission of the "restart" control message.

Operator intervention at both sender and receiver sites may be required.

Schematic Example
of
Sending "><CTLCKPnn" Control Messages

MCS COBOL

```
DATA DIVISION.
  01 MESSAGE.
    02 MES1 PIC X(8).
       VALUE "><CTLCKP".
    02 MES2 PIC XX.

COMMUNICATION SECTION.
  CD CD-OUT OUTPUT
    only user-initialized CD-output parameters are shown
    DESTINATION COUNT  COUNT-OUT
    TEXT LENGTH        LENGTH-OUT
    DESTINATION        DESTINATION-OUT.

PROCEDURE DIVISION.
  MOVE 1 TO COUNT-OUT.
  MOVE 10 TO LENGTH-OUT.
  MOVE destination TO DESTINATION-OUT.
  dynamic updating of current checkpoint nn by the user
  MOVE nn TO MES2.
  SEND CD-OUT FROM MESSAGE WITH {EMI|EGI}.
```

GPL

```
$H_CD OUTPUT, PREFIX='USER_';
only user-initialized CD-output parameters are shown
USER_DESTINATION_COUNT=1;
USER_TEXT_LENGTH=10;
USER_QUEUE_NAME="destination";
 :
DCL MESSAGE CHAR(10) INIT("><CTLCKP00");

dynamic updating of current checkpoint nn by the user

SUBSTR (MESSAGE 9,2)=nn;
$H_SEND 'ADDR(USER_OUTPUT_CD)', INADDR='ADDR(MESSAGE)',
        ENDCHAR=EMI;
 :
$H_SEND 'ADDR(USER_OUTPUT_CD)', INADDR='ADDR(MESSAGE)',
        ENDCHAR=EGI;
```

Example
of
Taking Checkpoints

MCS COBOL

```
DATA DIVISION.
  01 MODE COMP-2.
  01 CKINF PIC X(32).

PROCEDURE DIVISION.
  CALL "H_CK_UCHKPT" USING MODE CKINF.
```

GPL

```
DCL VARIABLE1 FIXED BIN(31);
DCL VARIABLE2 CHAR(32);
$H_CHKPT MODE=VARIABLE1, CKINF=VARIABLE2;
```

Example
of
Setting STATUS

MCS COBOL

```
DATA DIVISION.
  01 STATUS COMP-1.

PROCEDURE DIVISION.
  MOVE 10000 TO STATUS.
  CALL "H_CBL_USETST" USING STATUS.
```

GPL

use either form of coding

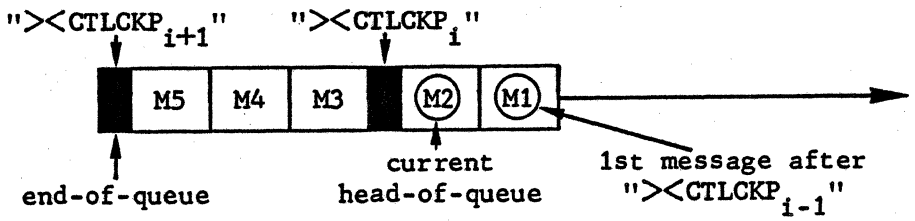
```
$H_SETST 10000;
```

```
DCL STATUS FIXED BIN(15);
STATUS=10000;
$H_SETST STATUS;
```

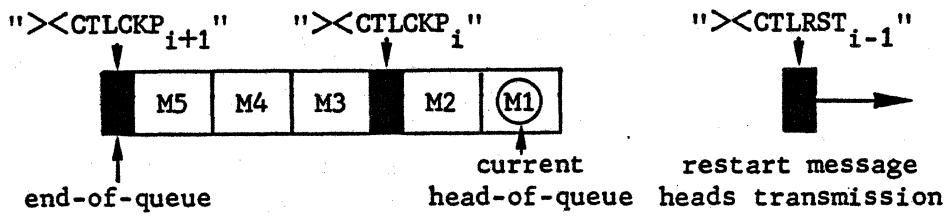
Recovery on the Emission Side

Sender Application Absent

Situation at Failure

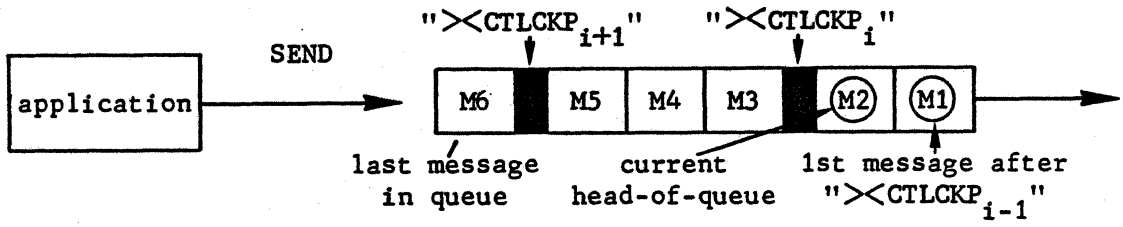


Situation at Restart

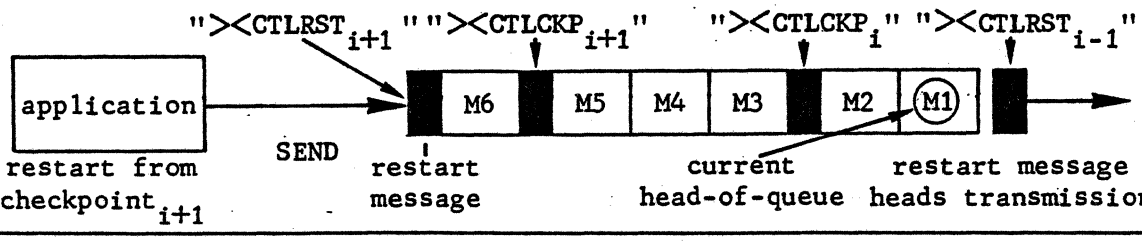


Sender Application Present

Situation at Failure



Situation at Restart



Recovery on the Reception Side

Recovery of transmission on the reception side involves,

- taking checkpoints
- receiving "restart" control messages
- restarting the application.

TAKING CHECKPOINTS

Whenever the receiver application receives the control message ">><CTLCKPnn" from the sender application, it takes its own checkpoint.

The checkpoint taken by the receiver application has the same sequence number as the checkpoint from the sender application that triggered it.

No control message is generated by the receiver application, instead, the checkpoint serves only as a recovery mark.

RECEIVING "RESTART" CONTROL MESSAGES

"Restart" is indicated by the control message ">><CTLRSTii" sent by the sender application to the receiver application.

The receiver application, in order to determine whether "restart" is possible or not, will perform the following decisions on the sequence number "ii" of the "restart" control message, namely,

- if ii ≠ nn, where nn is the sequence number of the last checkpoint taken by the receiver application, then the control message ">><CTLCKPnn+1" must have been lost along with its accompanying messages in the stream of data flow.

  In this case, no recovery is possible, and transmission must be reinitiated from the very beginning.

- if ii = nn, then the receiver application must resume from its current checkpoint.

  To do this, the application must proceed as follows,

  - in MCS COBOL, by the program coding,
    - issuing a call to the H_CBL_USETST run-time package to set the STATUS register to an abnormally high value, say 10,000
    - issuing a STOP RUN to return control immediately to the system.
  - in GPL, by the program coding,
    - issuing a H_SETST system primitive to set the STATUS register to an abnormally high value, say 10,000
    - issuing a RETURN to return control immediately to the system.

  The receiver application will terminate abnormally, and the system operator will then have to restart the application.

RESTARTING THE APPLICATION

When it has been determined that "restart" is possible, then the receiver application is restarted from its previous checkpoint.

"Restart" does not affect the sender application.

The following events take place on system restart after a step abort or system crash, namely,

- the system will reset the receiver application and its program-queues to their previous checkpoint state
- the terminal-queue related to the BSC link is empty
- the line is reactivated by the network control command RT LNnn, where nn specifies the line.

Operator intervention at both sites may be required to reactivate the line.

EXECUTING MCS APPLICATIONS

Preparing and executing communications applications involve

- preallocating a disk queue file
- linking MCS applications
- executing the communications step
- optimizing MCS applications.

Preallocating a Disk Queue File

If queues are to be held on disk, a file must be preallocated on a disk. Unless otherwise stated in the QUEUE command at CNC generation, disk queueing is the default option.

Preallocating a disk file is performed

- by using the file level utility PREALLOC
- before the CNC utility is executed, since the disk queue file must be preformatted as a result of the network generation.

If the file is to be modified, for example, altering the file size, the following procedure is performed in the sequence shown,

- the file is deallocated by
  - either using the file level utility DEALLOC
  - or declaring MAM=NO at system initialization
- the file is then preallocated through PREALLOC
- the CNC utility is then executed to update the system tables with the new file information.

The disk file cannot be deallocated

- if it is already defined for a network currently in use
- if MAM=YES or MAM=REFORMAT was declared at system initialization, thereby allocating the file to a system process group.

The file level utilities PREALLOC and DEALLOC are to be found in the Data Management Utilities manual.

Example
of
Preallocating a Disk Queue File

```
$JOB  ALLOCDQ  USER=UNAME  PROJECT=WAGE;
 PREALLOC  DQUEUE  EXPDATE=365  DEVCLASS=MS/M452
           GLOBAL=(MEDIA=VOL1  SIZE=5)
           BFAS=(SEQ=(BLKSIZE=500  RECSIZE=500));
$ENDJOB;
```

Syntax concerning the parameters in the file level utility PREALLOC :

- the external-file-name DQUEUE must be specified in a $ASSIGN statement at CNC execution
- the size of the file specified in the example as 5 cylinders is subject to the limits of the device class, see Network Generation manual
- a disk queue file must be a single-volume file for which the starting cylinder cannot be specified, hence GLOBAL must be used
- the parameter group BFAS=(SEQ=(BLKSIZE=500 RECSIZE=500)) must be specified as indicated.
- the values of BLKSIZE and RECSIZE will be overridden by CNC with values specified for QDBLKSZ in the GENCOM command.

Linking MCS Applications

An MCS application is compiled as follows,

- in MCS COBOL, by the $COBOL statement, see COBOL User Guide
- in GPL, by the $MACPROC and $GPL statements, see GPL User Guide.

The compile units, cu's, of the MCS application are then linked by the LINKER statement to form communications load-modules.

In "Outline of JCL Statements for Linking an MCS Application", the syntax for the JCL is as follows,

- $LIB statement (for details of parameters, see Library Maintenance Ref. Man.)
  - specifies the system library SYS.HCULIB containing the MAM run-time package
  - specifies the libraries containing the user cu's to be linked
  - describes the search path to the Static Linker.
- $LINKER statement (for details of parameters, see Linker manual)
  - OUTLIB specifies the library in which the load-module is to be stored.

    If the argument TEMP is chosen, then linking and load-module execution takes place within the same job.
  - parameters applicable to a monoprocess load-module :
    - ENTRY specifies the entry point to the application.

      The default entry point is the name of the load-module.
    - COMFAC identifies the application as a monoprocess load-module.
  - parameters applicable to a multiprocess load-module :
    - COMFILE names the input enclosure required to link a multiprocess load-module.
- input enclosure statements :

  The information given applies in the case where the system procedure STUSERS is used.

  An MCS application written in GPL has the choice of either using the system procedure STUSERS or a user-defined ENTRY procedure, see "Example of user-defined ENTRY Procedure", page 2-49.
  - ENTRY and LINKTYPE must be entered exactly as indicated with the parameters given
  - TASK statement :
    - USERn is used by MCS to start the indicated task in the order from USER1 through USER6, where appropriate
    - START=STUSERn indicates the entry point of the associated task USERn
  - REPLACE statement :
    - cu-namen is the entry point in the application to be linked to the task USERn
    - CU=STUSERn indicates the compile-unit of the associated task USERn.

Outline of JCL Statements
for
Linking an MCS Application

```
$JOB  job-name  USER=user-name  PROJECT=project-name;
 LIB  CU  INLIB1={ SYS.HCULIB                  }
                 { TEMP                        }
                 { (simple-file-description)   }
          INLIB2={ TEMP                        }
                 { (simple-file-description)   }
            :
        [INLIB5={ TEMP                        }]
                 { (simple-file-description)   };
 LINKER  load-module-name  [ENTRY=cu-name]
        [OUTLIB={ TEMP                        }]
                 { (simple-file-description)   }
        {COMFAC                               }
        {COMFILE=*input-enclosure-name        };
```

The following input enclosure is to link a multiprocess load-module where COMFILE is specified.

The enclosure is bypassed where COMFAC is specified.

```
$INPUT  input-enclosure-name;
 ENTRY=STUSERS,
 LINKTYPE=BMAM,
 TASK=(USER1  START=STUSER1),
 REPLACE=(DUMMY  cu-name1  CU=STUSER1),
 TASK=(USER2  START=STUSER2),
 REPLACE=(DUMMY  cu-name2  CU=STUSER2),
    :
 TASK=(USER6  START=STUSER6),
 REPLACE=(DUMMY  cu-name6  CU=STUSER6),
$ENDINPUT ;
```

```
$ENDJOB;
```

LINKING A MONOPROCESS MCS LOAD-MODULE

In "Example of Linking a Monoprocess Load-Module", the syntax for the JCL is as follows,

- $LIB statement :
  - UCULIB is the cu-library on volume VOL1 containing the user application.
- $LINKER statement :
  - MAM1 is the name of the load-module and is also the entry point to the user application
  - ULMLIB is the user load-module library on the volume VOL1 in which the load-module is to be stored.

LINKING A MULTIPROCESS MCS LOAD-MODULE

Linking a multiprocess MCS load-module can be done

- either by using the system procedure STUSERS
- or, in the case of a GPL, by creating a user-defined ENTRY procedure and using it in place of STUSERS.

° Using the System Procedure STUSERS :

In "Example of Linking a Multiprocess Load-Module", the syntax for the JCL is as follows,

- $LIB statement :
  - UCULIB1 and UCULIB2 are the cu-libraries on the volume VOL1 containing the user application.
- $LINKER statement :
  - ULMLIB is the user load-module library on the volume VOL1 in which the multiprocess load-module MAM2 is to be stored.
- input enclosure statements (sequence of statements is not significant)
  - DATCOLL is an entry point to the user application which is to be linked to the task USER2
  - INQRESP is an entry point to the user application which is to be linked to the task USER1.

Examples of Linking MCS Applications

Linking a Monoprocess Load-Module

```
$JOB  LINK1  USER=UNAME  PROJECT=ACCT;
 LIB  CU  INLIB1=SYS.HCULIB
          INLIB2=(UCULIB  DEVCLASS=MS/M452  MEDIA=VOL1);
 LINKER  MAM1
         OUTLIB=(ULMLIB  DEVCLASS=MS/M452  MEDIA=VOL1)
         COMFAC;
$ENDJOB;
```

Linking a Multiprocess Load-Module

```
$JOB  LINK2  USER=UNAME  PROJECT=WAGE;
 LIB  CU  INLIB1=SYS.HCULIB
          INLIB2=(UCULIB1  DEVCLASS=MS/M452  MEDIA=VOL1)
          INLIB3=(UCULIB2  DEVCLASS=MS/M452  MEDIA=VOL2);
 LINKER  MAM2
         OUTLIB=(ULMLIB  DEVCLASS=MS/M452  MEDIA=VOL1)
         COMFILE=*LINK;
$INPUT  LINK;
 ENTRY=STUSERS,
 LINKTYPE=BMAM,
 TASK=(USER1  START=STUSER1),
 TASK=(USER2  START=STUSER2),
 REPLACE=(DUMMY  DATCOLL  CU=STUSER2),
 REPLACE=(DUMMY  INQRESP  CU=STUSER1),
$ENDINPUT;
$ENDJOB;
```

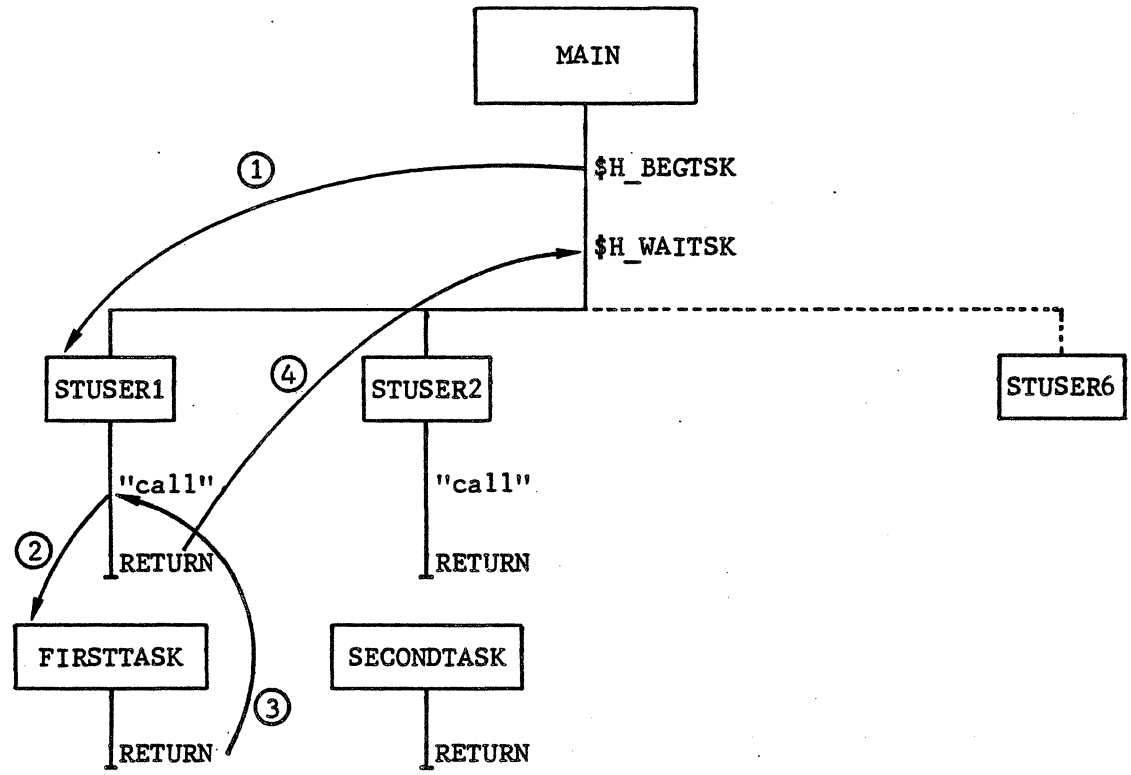
LINKING A MULTIPROCESS LOAD-MODULE (continued)

° Using a user-defined ENTRY Procedure :

In GPL, the user has the choice of specifying and managing his own tasks and subtasks by the means of the H_BEGTSK and H_WAITSK primitives, whereby

- the name of the procedure in the ENTRY command is user-defined
- the names of the tasks in the TASK commands are user-defined
- the entry points, however, invoked by the above primitives are referenced by STUSERn, n ranging from 1 through 6, such that
  - each STUSERn executes a "call" to the user application through a user defined entry point specified at linkage time
  - each STUSERn invokes MCS to perform housekeeping functions before terminating the associated task.

An additional advantage of the user-defined ENTRY procedure is that other tasks not containing MCS primitives can be included.



The secondary tasks identified by FIRSTTASK and SECONDTASK are user-defined and contain MCS primitives.

1 $H_BEGTSK directive is issued to STUSER1

2 This results in a "call" to the user-defined secondary task FIRSTTASK

3 After FIRSTTASK is performed, control is returned to STUSER1

4 MAM is now invoked to perform housekeeping functions.

Example of
user-defined ENTRY Procedure
(the name of the user-defined procedure is MAIN)

Example of Programming
MAIN in GPL

```
MAIN:PROC;
  :
  :
 DCL  1  STRUC1 ... ;
 DCL  1  STRUC2 ... ;
  :
  :
$H_BEGTSK  NAME="FIRSTTASK"    OPTIONS=STRUC1  ;
$H_BEGTSK  NAME="SECONDTASK"   OPTIONS=STRUC2  ;
$H_WAITSK  NAME="FIRSTTASK";
$H_WAITSK  NAME="SECONDTASK";
RETURN;
END;
```

up to 6 "tasks" may be declared

Linking Multiprocess Load-Module
using MAIN

```
$JOB  LINK2  USER=UNAME  PROJECT=WAGE;
 LIB  CU  INLIB1=SYS.HCULIB
          INLIB2=(UCULIB1  DEVCLASS=MS/M452  MEDIA=VOL1)
          INLIB3=(UCULIB2  DEVCLASS=MS/M452  MEDIA=VOL2);
 LINKER  MAM2  OUTLIB=(ULMLIB  DEVCLASS=MS/M452  MEDIA=VOL1)
         COMFILE=*LINK;
$INPUT  LINK;
 ENTRY=MAIN,
 LINKTYPE=BMAM,
 TASK=(FIRSTTASK  START=STUSER1),
 TASK=(SECONDTASK  START=STUSER2),
 REPLACE=(DUMMY  DATCOLL  CU=STUSER2),
 REPLACE=(DUMMY  INQRESP  CU=STUSER1),
$ENDINPUT;
$ENDJOB;
```

Format and Syntax of $QASSIGN Statement

```
QASSIGN  symbolic-queue-name  [. symbolic-subqueue1-name
                               [. symbolic-subqueue2-name
                               [. symbolic-subqueue3-name]]]

         external-queue1-name

         [ { IN    [ACCESS = {LIN}] } ]   [ REPLY = external-queue2-name ]
         [ { INOUT [         {RR }] } ]
         [ { OUT                    } ]

         [ SITE = system-name ] ;
```

Description of Parameters

external-queue-name
- ranges from 1 through 12 alphanumeric characters and is the name of the corresponding QUEUE command at network generation.

symbolic-queue-name
- identifies the logical queue defined as follows
  - in MCS COBOL, in data-name-1 of the CD area (input and output)
  - in GPL, by QUEUE_NAME of H_CDOUT or H_CDIN.

symbolic-subqueue-name
- applicable only for input queues for partitioning the logical queue as follows
  - in MCS COBOL, in data-name-2, data-name-3 & data-name-4 of the input CD area
  - in GPL, by SUBQUEUE_NAME, SUBQUEUE2_NAME and SUBQUEUE3_NAME of H_CDIN.

ACCESS
- applicable only to input queues if partitioned into subqueues, to specify how the subqueues are to be scanned on "receive" as follows
  - LIN : "linear", the search starts at the first subqueue specified, default
  - RR : "round-robin", the search resumes at the subqueue immediately following the subqueue that provided data on the last "receive".

IN
- "input mode", applicable only to program-queues, default.

INOUT
- "input/output mode", applicable only to program-queues and must not be specified for subqueues.

OUT
- "output mode", must not be specified for subqueues.

REPLY
- applicable only to "application-to-application" communication and must not be specified with the IN "processing mode".

SITE
- identifies a DSA system which can be either "lsys" or "rsys" being DPS 7's.

DEFINITION OF $QASSIGN

The $QASSIGN statement performs the following,

- defines the symbolic queues and, where applicable, the subqueues to be used on input
- establishes the correspondence between the symbolic queue and its associated subqueue(s), if the queue is to be used for input, and the external queue specified in the QUEUE command at network generation
- identifies the "processing mode" of the queue with respect to "send's" and "receive's"
- allocates the queues to the job step.

RULES FOR USING $QASSIGN

The following rules govern the use of the $QASSIGN statement,

- the order of the list of subqueues within a symbolic input queue is determined by the order of $QASSIGN statements within the step enclosure
- the maximum number of $QASSIGN statements permitted within the step enclosure is 26
- the ACCESS parameter is only relevant in the first $QASSIGN statement defining a queue structure, that is, subqueues within the queue
- a $QASSIGN statement is mandatory for each input queue associated with the application
- $QASSIGN statements for output queues are optional
- REPLY defines another "program-queue" which is identified as the source of input messages in the destination application's input CD area (H_CDIN), such that the destination application can answer back, using in its turn, the "program-queue" thus defined as the "destination"
- on output queues where the REPLY option is not specified, an implicit REPLY (towards the first input queue among the $QASSIGN statements) is provided if any such queue exists
- where disk queues are used, no $ASSIGN statement is required because the file is opened at system level and is therefore sharable by all process groups.

Example of
Run-time JCL
with $QASSIGNs

```
$JOB  TELECOM  USER=UNAME  PROJECT=ACCT  CLASS=H;
 STEP  MAM3  (ULMLIB  DEVCLASS=MS/M452  MEDIA=VOL1);

  a : defines INQ queue structure,   b : defines terminal queues

        a  { QASSIGN  INQ.SUB1  PRG1  ACCESS=RR;
           { QASSIGN  INQ.SUB2  PRG2;
           { QASSIGN  INQ.SUB3  PRG3;
           { QASSIGN  INQ.SUB4  PRG4;
        b  { QASSIGN  OUTQ  TRM1  OUT;
           { QASSIGN  OUTW  TRM2  OUT;

 ENDSTEP;
$ENDJOB;
```

EXAMPLE OF RUN-TIME JCL WITH $QASSIGNS

In "Example of Run-time JCL with $QASSIGNs", the syntax of the statements is as follows,

- $JOB statement :
  - CLASS=H is recommended for a communications step to ensure adequate response time in a multiprogramming environment
- $STEP statement :
  - MAM3 is the multiprocess load-module linked and stored in a previous session in the user load-module library ULMLIB
- $QASSIGN statement :
  - group a , each symbolic-subqueue corresponds to a unique external-queue-name, that is, SUBn maps on to PRGn, specified in the associated QUEUE commands at CNC generation, and will be scanned in "round-robin" in the order specified
  - group b , the symbolic-output-queues OUTQ and OUTW are associated with their respective terminal-queues TRM1 and TRM2.

Optimizing MCS Applications

General recommendations for optimizing MCS applications involve the use of the "timer" which is set accordingly as follows,

- in MCS COBOL, by the statement CALL to the run-time package H_TM_USETTM
- in GPL, by the system primitive $H_SETELT.

The "timer" is set under the following conditions,

- when issuing a RECEIVE (H_RECEIVE) with "no data"
- when issuing an ACCEPT (H_MSGCNT)
- when handling several program-queues.

ISSUING RECEIVE (H_RECEIVE)

The 2 conditions for issuing a RECEIVE (H_RECEIVE) are,

- when not qualified by a "no data" clause
- when specified with a "no data" clause.

° Not Qualified by a "No Data" Clause :

One of the following 2 conditions will occur,

- if at least 1 message is queued, then it will be delivered into the user-specified working area, before control is given back to the application to determine what next to do
- if the queue is empty, MCS suspends the application until a message arrives in the queue.

  When a message arrives, it is delivered into the user-specified working area, and control is again returned to the application.

° Specified with a "No Data" Clause :

RECEIVE (H_RECEIVE) with "no data" is used to scan a queue while executing another process or while awaiting the occurrence of other events.

If a message is available in the queue, the "no data" condition is bypassed and the resulting action is the same as for a RECEIVE (H_RECEIVE).

If no message is available in the queue, the "no data" condition is entered and the application will loop on this condition until a message arrives, thereby causing the application

- to occupy all the CPU time not used by BTNS, FNPS or GCOS
- to prevent the execution of any batch program
- to reduce the throughput of the system.

In order to avoid the degradation in system performance, setting the "timer" in the "no data" loop allows the application to be suspended for a user-specified time before being re-activated.

Schematic Example
of
RECEIVE with "no data"

MCS COBOL

```
DATA DIVISION.
  01 BUFFER PIC X(nn).
COMMUNICATION SECTION.
  CD CD-IN INPUT
    TEXT LENGTH    LENGTH-IN
PROCEDURE DIVISION.
  MOVE nn TO LENGTH-IN.
  RECV.
    RECEIVE CD-IN MESSAGE INTO BUFFER NO DATA GO TO SETTIMER.
  SETTIMER.
    set timer
    GO TO RECV.
```

GPL

```
  $H_CD INPUT PREFIX='FIRST_';
  DCL BUFFER CHAR(nn);
  RECV:;
    $H_RECEIVE 'ADDR(FIRST_INPUT_CD)', OUTADDR='ADDR(BUFFER)',
               LENGTH=nn, NWAIT;
    IF $H_TESTRC EMPTY; THEN GO TO SETTIMER;
  SETTIMER:;
    set timer
    GO TO RECV;
```

Example of
Setting Timer
(nn is specified in milliseconds)

MCS COBOL

```
DATA DIVISION.
  01 DELAY-TIME COMP-2.
PROCEDURE DIVISION.
  MOVE nn TO DELAY-TIME.
  CALL "H_TM_USETTM" USING DELAY-TIME.
```

GPL

```
DCL DELAY_TIME FIXED BIN(31);
DELAY_TIME=nn;
$H_SETELT MILSEC=DELAY_TIME;
```

ISSUING ACCEPT (H_MSGCNT)

ACCEPT (H_MSGCNT) is used to ascertain the number of messages in a queue while executing another process or while awaiting the occurrence of other events.

If the message count is not zero, that is, there is at least 1 message in the queue, the application will continue normal processing.

If the message count is zero, the application will loop on this condition until a message arrives in the queue to update the message count to non-zero, thereby causing the application

- to occupy all the CPU time not used by BTNS, FNPS or GCOS
- to prevent the execution of any batch program
- to reduce the throughput of the system.

In order to avoid the degradation in system performance, setting the "timer" in the "message count = zero" loop allows the application to be suspended for a user-specified time before being re-activated.

Example of
Setting Timer with
ACCEPT (H_MSGCNT)

MCS COBOL

```
DATA DIVISION.
  01 DELAY-TIME COMP-2.
COMMUNICATION SECTION.
  CD CD-IN INPUT
    MESSAGE COUNT      MSG-CNT
PROCEDURE DIVISION.
  ACCPT.
    ACCEPT CD-IN MESSAGE COUNT.
    IF MSG-CNT=0 GO TO SETTIMER.
  SETTIMER.
    MOVE nn TO DELAY-TIME.
    CALL "H_TM_USETTM" USING DELAY-TIME.
    GO TO ACCPT.
```

GPL

```
$H_CD INPUT PREFIX='FIRST_';
DCL DELAY_TIME FIXED BIN(31);

ACCPT:;
  $H_MSGCNT 'ADDR(FIRST_INPUT_CD)';
  IF FIRST_MSG_CNT=0 THEN GO TO SETTIMER;
SETTIMER:;
  DELAY_TIME=nn;
  $H_SETELT MILSEC=DELAY_TIME;
  GO TO ACCPT;
```

HANDLING SEVERAL PROGRAM-QUEUES

As a general rule, most applications handle only 1 program-queue which is the "point-of-entry" for all data received.

In some cases, the application may have to handle several program-queues under the following conditions,

- an order of priority is to be established between more than 1 source of messages so that 1 particular source can be selected at a given time
- the application is to communicate with a complex of terminals, and with either or both
  - a process of the same application
  - another application.

Problem : How to scan 3 program-queues Q1, Q2 and Q3 in which an application is to receive messages.

Solution 1 : Each of the 3 queues is defined by a separate $QASSIGN statement, whereby each is unrelated to the others.

The application can then establish the order of priority in which the queues are to be scanned, say, Q1 having the lowest priority and Q3, the highest, by performing the following,

- looping on RECEIVE (H_RECEIVE) with "no data" successively for each queue starting with Q3, since Q3 has the highest priority
- setting the "timer" in the "no data" loop to suspend the application until a message eventually becomes available in the queue concerned.

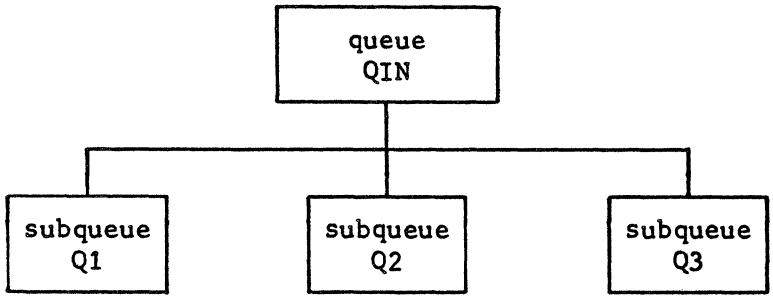
The drawbacks of this solution are,

- by establishing priority among the queues, the application is unnecessarily suspended if, say, Q3 which has the highest priority is empty, and pending the arrival of a message in Q3, the other queues Q1 and Q2 are left unscanned.

  This unnecessary suspension can be avoided if all the queues are scanned and if no message is present in any of the queues, to set the "timer" before rescanning the queues.

- the "timer" is arbitrarily set and therefore is difficult to optimize, giving rise to the following situations,
  - if the "wait-time" is too large, the "turn-around" time for processing becomes too slow
  - if the "wait-time" is too small, the application will loop on itself until a message becomes available in the queue concerned, thereby defeating the purpose of the "timer".

Solution 2 : Each of the 3 queues can be defined as subqueues of the queue, say, QIN, such that the $QASSIGN statements for the 3 queues represent an interrelated hierarchy, as shown in the diagram following,



In this case, the application has only to issue a RECEIVE (H_RECEIVE) not qualified by a "no data" clause to the queue QIN, without having to specify the individual subqueues Q1, Q2 and Q3.

As will be seen later on, the priority to be given to the subqueue is resolved when the "received" message is to be processed.

The advantages of this solution are,

- from the point of view of the scanning mechanism, the 3 queues Q1, Q2 and Q3 represent a single queue, QIN, and the likelihood for all 3 subqueues to be empty at the same time is small and therefore there is no need to specify the "no data" condition
- if no message is available in any of the 3 subqueues, MAM will suspend the application, and processing will resume only as soon as a message arrives in any of the subqueues.
- the RECEIVE (H_RECEIVE) will indicate in the input CD (H_CD) from which subqueue the message has been "received". The application can then place whatever priority it wants on processing the message. In this case, for consistency, the messages from subqueue Q3 will be processed first, even if there are other messages preceding it from the subqueues Q1 and Q2.

The method of scanning the queues is dealt with in the description of the ACCESS parameter of the $QASSIGN statement on page 2-51.

SECTION III

MCS DATA FORMATS

This section deals with the format of data sent to and received from a terminal by an MCS application.

The data is held in the working area defined as the user buffer and accessed by the SEND (H_SEND) and RECEIVE (H_RECEIVE) communications verbs.

Data comprises,

- graphic symbols, which are,
  - numeric characters
  - alphabetic characters, both upper and lower case
  - special characters, such as, punctuation, mathematical and currency symbols
- control codes, which provide terminal management functions, such as,
  - editing or service functions, for example, carriage-return, tabulation and cursor positioning
  - auxiliary device commands, such as, to the cassette handler
  - timing functions, such as, "time-fill's" to allow for slow mechanical functions on a printer
  - message delimiters, such as, VIP headers, "end-of-transmission-block", and trailers.

This section is dealt with in terms of,

- symbolic representation
- transmission modes
- MCS data processing
- data representation.

SYMBOLIC REPRESENTATION

On keyboards where the graphic symbol or control code is present, pressing the respective key results in the transmission of a 1-byte hexadecimal value from the terminal to the user buffer.

However, not all terminals are equipped with the set of graphic symbols and control codes as shown in the "ASCII" and "EBCDIC" tables.

In the absence of such graphic symbols and control codes, another means must be found to represent these, namely, by using the graphic symbols >< to denote

- the character-encoding mark ><Cxy, where xy represents 2 hexadecimal-digits corresponding to the EBCDIC value of the graphic symbol or control code
- the mark form representation ><ccc, where ccc represents the 3-character mnemonic of the control code, see "Control Codes"
- the repeat mark ><Rab, where ab represents 2 decimal-digits, being the number of times the graphic symbol, and in some cases, the control code, is to be repeated
- the VIP header of the form ><U03 preceding the "status" and "function-code's".

This symbolic representation of data is a feature provided by MCS at user program level.

In the "ASCII" table, values greater than hexadecimal 80, are not represented either by control codes or graphic symbols.

These "unreserved" codes can therefore only be symbolically represented, and when specified in unedited mode, both on input and output, can be used for specific user-defined functions.

CONTROL CODES

List of Common Control Codes

(for specific control codes, see Section VI "Programming Terminals")

COBOL denotes "collating sequence" abbreviated to COBOL C/S in tables

control code	EBCDIC	ASCII	COBOL
ACK acknowledge	2E	06	47
BO1 advance 1 page = FF▽	0C	0C	13
BO3 advance 1 line = CR▽ LF▽			
BEL bell	2F	07	48
for VIP terminals, BEL generates BLK			
BLK blink	5F	5E	96
BS▽ backspace	16	08	23
CAN cancel	18	18	25
CR▽ carriage return	0D	0D	14
DC1 device control 1	11	11	18
DC2 device control 2	12	12	19
for VIP terminals, DC2 generates forward space			
DC3 device control 3	13	13	20
DC4 device control 4	3C	14	61
DEL delete	07	7F	08
DLE data link escape	10	10	17
EM▽ end of medium	19	19	26
ENQ enquiry	2D	05	46
EOT end of transmission	37	04	56
ESC escape	27	1B	40

control code	EBCDIC	ASCII	COBOL
ETB end of transmission block	26	17	39
ETX end of text	03	03	04
FF▽ form feed = BO1	0C	0C	13
FS▽ file separator	1C	1C	29
GS▽ group separator	1D	1D	30
HT▽ horizontal tabulation	05	09	06
LF▽ line feed	25	0A	38
NAK negative acknowledge	3D	15	62
NL▽ new line (BSC3270) = CR▽ LF▽	15	85	22
NUL null	00	00	01
RS▽ record separator	1E	9D	21
SI▽ shift in	0F	0F	16
SO▽ shift out	0E	0E	15
SOH start of header	01	01	02
STX start of text	02	02	03
SUB substitute	3F	1A	64
SYN synchronous idle	32	16	51
US▽ unit separator	1F	1F	32
VT▽ vertical tabulation	0B	0B	12

Control Code in Mark Form

Where a touch-key representing a control code is absent on the keyboard, the only means of entering the control code is using its mnemonic form preceded by the symbols ><.

Using the table "Control Codes", which lists the common codes known by their mnemonic form to the "stream processor", the sequence ><ESC can be keyed in, if for example, the touch-key representing ESC is not present on the keyboard.

On output, however, control codes are treated according to their applicability to the line procedure, namely, in the output normal mode,

- those codes applicable on output are translated into 1-byte hexadecimal values
- non-applicable codes pass unchanged in their original mark form.

The following control codes in mark form pass unchanged to the terminal in output normal mode, and are deleted if preceded by the "repeat" function.

line procedure	control codes not applicable on output				
BSC2780 BSC3270 VIP	><ACK	><ENQ	><RS▽	><SOH	><SYN
	><CAN	><ETB	><SI▽	><STX	><US▽
	><DLE	><NAK	><SO▽	><SUB	><VT▽
TC/TTY	><BLK				

Character-Encoded Form

Character-encoding is used to enter 2 hexadecimal-digits preceded by the mark ><C, and is a means of representing such data as,

- control codes for which no touch-key representing their function exists on the keyboard
- graphic symbols which have no displayable equivalents on the submitter terminal but which can be displayed on the receiver terminal, see "Processing on Output".

In input normal and marked modes, the character-encoded mark ><C passes the following 2 hexadecimal-digits as a 1-byte hexadecimal value to the user buffer.

In output normal mode, character-encoding is used to represent control codes not applicable for output for the line procedure concerned, see the individual tables for the different line procedures under "Output Normal Mode".

The non-applicable control code, is treated like a non-displayable graphic symbol, and is translated from its 1-byte hexadecimal value into character-encoded form and transmitted to the terminal.

TRANSMISSION MODES

Except for BSC2780-like terminals, the transmission from a terminal to the Level 64 is in ASCII code.

The translation of the ASCII code to EBCDIC code, which is the internal code of the DPS 7, is performed by means of the TCT, by either the URP or by the DN7100, depending on whether BTNS or FNPS is involved.
BSC2780-like terminals transmit in EBCDIC code, so further translation is not required.

The EBCDIC code is interpreted by the "stream processor" of QMON for the type of conversion according to the transmission mode specified by the user.

The data, having been suitable treated, then passes to the user buffer for processing by the MCS application.

On output, the process in reverse operates. Transmissions to the terminal which correspond to neither a displayable graphic symbol nor to a valid control code, result in temporizing "time-fill's".
The mechanism of the "time-fill" is a useful feature when programming for non-standard terminals.

The transmission mode can be declared at network generation and dynamically changed during the course of the communications session.

For both input and output, normal and unedited transmission modes are available. However, for input, an additional transmission mode, the marked mode is available.

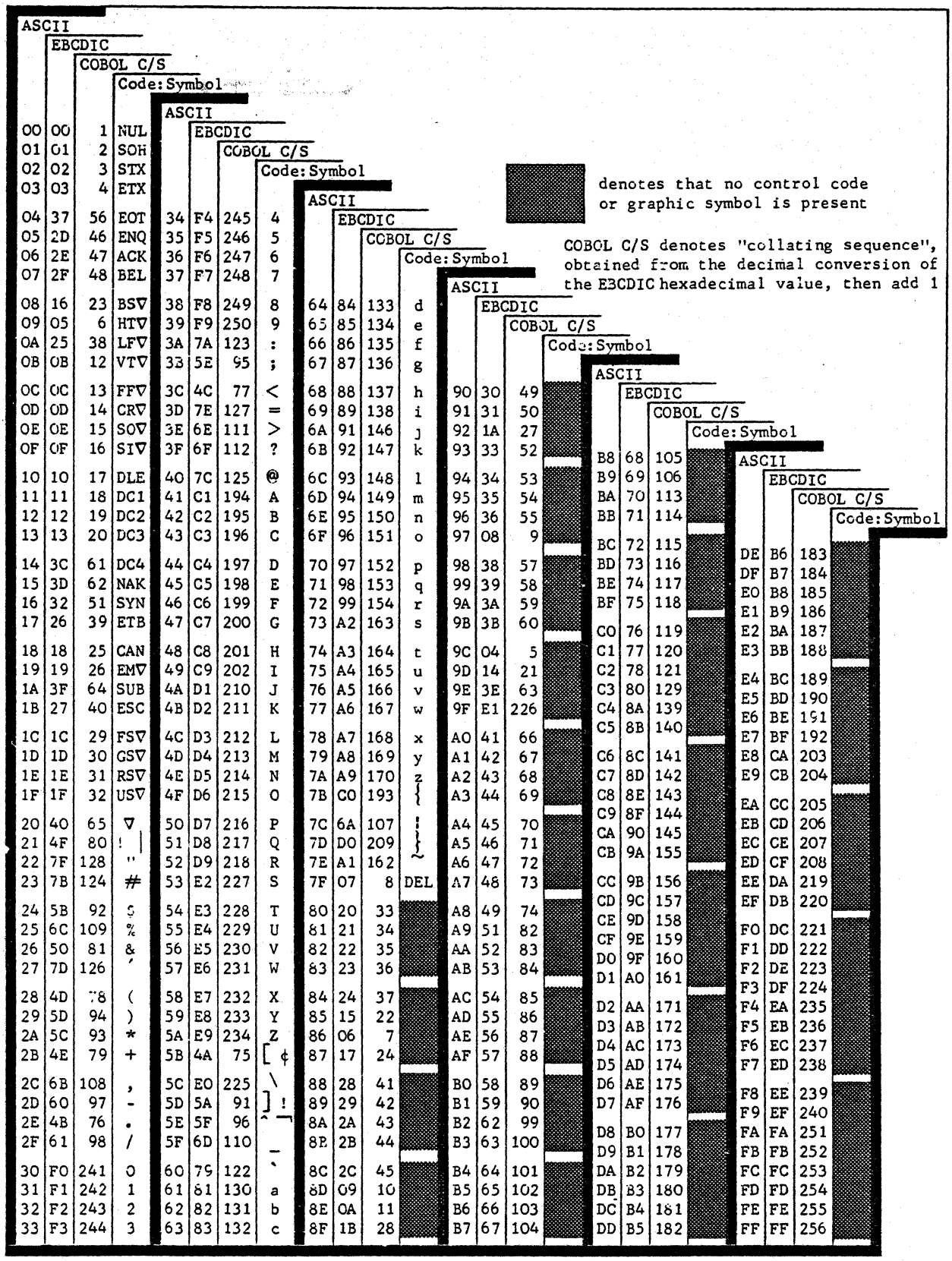
In the following examples of transmission modes, the conventions used are,

-  indicates a 1-byte EBCDIC hexadecimal value
-  indicates the transmission of a control code as a 1-byte hexadecimal value activated by the "carriage-return" control key
-  indicates the transmission of a graphic symbol as a 1-byte hexadecimal value activated by a touch-key, in this case, "D"
-  indicates the 1-byte EBCDIC hexadecimal value and is used to show the conversion from/to 2 hexadecimal-digits preceded by ><C
-  indicates any data not affected by the "stream processor"
- indicates the direction of transfer either from the terminal to the user buffer or vice versa
- indicates the string-grouping of characters as processed by the "stream processor"

For the normal output transmission mode, certain control codes and graphic symbols do not apply for specific line procedures.
"Output Normal Mode" tables are provided for each line procedure supported.

For input, the table "Input Marked Mode" treats all the common control codes in mark form and passes all other data unchanged to the Level 64.
In the input normal, single control codes, except for HTV and ESC, are deleted.

ASCII

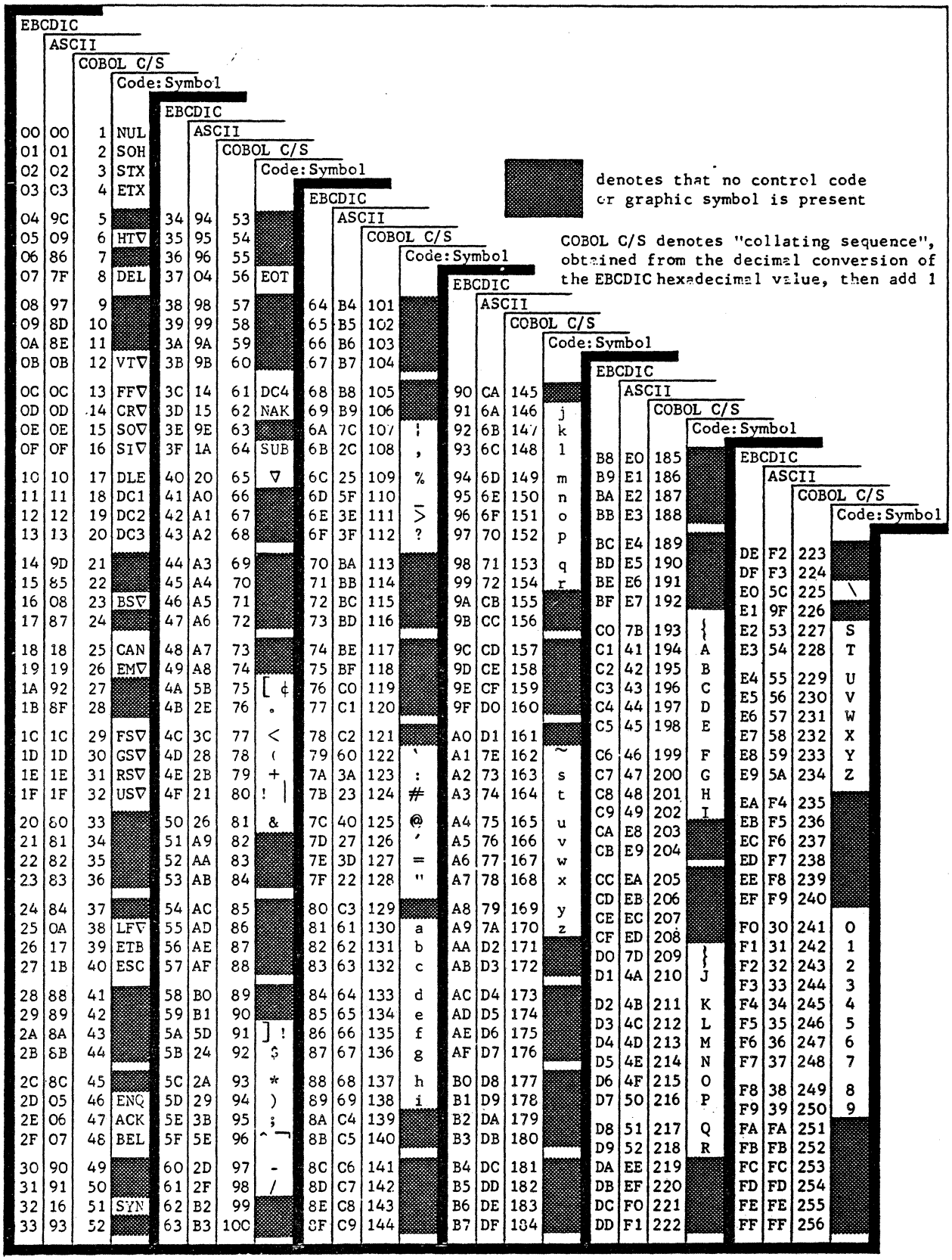


ASCII	EBCDIC	COBOL C/S	Code: Symbol
00	00	1	NUL
01	01	2	SOH
02	02	3	STX
03	03	4	ETX
04	37	56	EOT
05	2D	46	ENQ
06	2E	47	ACK
07	2F	48	BEL
08	16	23	BS∇
09	05	6	HT∇
0A	25	38	LF∇
0B	0B	12	VT∇
0C	0C	13	FF∇
0D	0D	14	CR∇
0E	0E	15	SO∇
0F	0F	16	SI∇
10	10	17	DLE
11	11	18	DC1
12	12	19	DC2
13	13	20	DC3
14	3C	61	DC4
15	3D	62	NAK
16	32	51	SYN
17	26	39	ETB
18	18	25	CAN
19	19	26	EM∇
1A	3F	64	SUB
1B	27	40	ESC
1C	1C	29	FS∇
1D	1D	30	GS∇
1E	1E	31	RS∇
1F	1F	32	US∇
20	40	65	∇
21	4F	80	! |
22	7F	128	"
23	7B	124	#
24	5B	92	$
25	6C	109	%
26	50	81	&
27	7D	126	'
28	4D	78	(
29	5D	94	)
2A	5C	93	*
2B	4E	79	+
2C	6B	108	,
2D	60	97	-
2E	4B	76	.
2F	61	98	/
30	F0	241	0
31	F1	242	1
32	F2	243	2
33	F3	244	3
34	F4	245	4
35	F5	246	5
36	F6	247	6
37	F7	248	7
38	F8	249	8
39	F9	250	9
3A	7A	123	:
3B	5E	95	;
3C	4C	77	<
3D	7E	127	=
3E	6E	111	>
3F	6F	112	?
40	7C	125	@
41	C1	194	A
42	C2	195	B
43	C3	196	C
44	C4	197	D
45	C5	198	E
46	C6	199	F
47	C7	200	G
48	C8	201	H
49	C9	202	I
4A	D1	210	J
4B	D2	211	K
4C	D3	212	L
4D	D4	213	M
4E	D5	214	N
4F	D6	215	O
50	D7	216	P
51	D8	217	Q
52	D9	218	R
53	E2	227	S
54	E3	228	T
55	E4	229	U
56	E5	230	V
57	E6	231	W
58	E7	232	X
59	E8	233	Y
5A	E9	234	Z
5B	4A	75	[¢
5C	E0	225	\
5D	5A	91	] !
5E	5F	96	^ ¬
5F	6D	110	_
60	79	122	`
61	81	130	a
62	82	131	b
63	83	132	c
64	84	133	d
65	85	134	e
66	86	135	f
67	87	136	g
68	88	137	h
69	89	138	i
6A	91	146	j
6B	92	147	k
6C	93	148	l
6D	94	149	m
6E	95	150	n
6F	96	151	o
70	97	152	p
71	98	153	q
72	99	154	r
73	A2	163	s
74	A3	164	t
75	A4	165	u
76	A5	166	v
77	A6	167	w
78	A7	168	x
79	A8	169	y
7A	A9	170	z
7B	C0	193	{
7C	6A	107	¦
7D	D0	209	} ~
7E	A1	162	
7F	07	8	DEL
80	20	33	
81	21	34	
82	22	35	
83	23	36	
84	24	37	
85	15	22	
86	06	7	
87	17	24	
88	28	41	
89	29	42	
8A	2A	43	
8B	2B	44	
8C	2C	45	
8D	09	10	
8E	0A	11	
8F	1B	28	
90	30	49	
91	31	50	
92	1A	27	
93	33	52	
94	34	53	
95	35	54	
96	36	55	
97	08	9	
98	38	57	
99	39	58	
9A	3A	59	
9B	3B	60	
9C	04	5	
9D	14	21	
9E	3E	63	
9F	E1	226	
A0	41	66	
A1	42	67	
A2	43	68	
A3	44	69	
A4	45	70	
A5	46	71	
A6	47	72	
A7	48	73	
A8	49	74	
A9	51	82	
AA	52	83	
AB	53	84	
AC	54	85	
AD	55	86	
AE	56	87	
AF	57	88	
B0	58	89	
B1	59	90	
B2	62	99	
B3	63	100	
B4	64	101	
B5	65	102	
B6	66	103	
B7	67	104	
B8	68	105	
B9	69	106	
BA	70	113	
BB	71	114	
BC	72	115	
BD	73	116	
BE	74	117	
BF	75	118	
C0	76	119	
C1	77	120	
C2	78	121	
C3	80	129	
C4	8A	139	
C5	8B	140	
C6	8C	141	
C7	8D	142	
C8	8E	143	
C9	8F	144	
CA	90	145	
CB	9A	155	
CC	9B	156	
CD	9C	157	
CE	9D	158	
CF	9E	159	
D0	9F	160	
D1	A0	161	
D2	AA	171	
D3	AB	172	
D4	AC	173	
D5	AD	174	
D6	AE	175	
D7	AF	176	
D8	B0	177	
D9	B1	178	
DA	B2	179	
DB	B3	180	
DC	B4	181	
DD	B5	182	
DE	B6	183	
DF	B7	184	
E0	B8	185	
E1	B9	186	
E2	BA	187	
E3	BB	188	
E4	BC	189	
E5	BD	190	
E6	BE	191	
E7	BF	192	
E8	CA	203	
E9	CB	204	
EA	CC	205	
EB	CD	206	
EC	CE	207	
ED	CF	208	
EE	DA	219	
EF	DB	220	
F0	DC	221	
F1	DD	222	
F2	DE	223	
F3	DF	224	
F4	EA	235	
F5	EB	236	
F6	EC	237	
F7	ED	238	
F8	EE	239	
F9	EF	240	
FA	FA	251	
FB	FB	252	
FC	FC	253	
FD	FD	254	
FE	FE	255	
FF	FF	256	

denotes that no control code or graphic symbol is present

COBOL C/S denotes "collating sequence", obtained from the decimal conversion of the EBCDIC hexadecimal value, then add 1

EBCDIC



EBCDIC	ASCII	COBOL C/S	Code:Symbol
00	00	1	NUL
01	01	2	SOH
02	02	3	STX
03	C3	4	ETX
04	9C	5	
05	09	6	HTV
06	86	7	
07	7F	8	DEL
08	97	9	
09	8D	10	
0A	8E	11	
0B	0B	12	VTV
0C	0C	13	FFV
0D	0D	14	CRV
0E	0E	15	SOV
0F	0F	16	SIV
10	10	17	DLE
11	11	18	DC1
12	12	19	DC2
13	13	20	DC3
14	9D	21	
15	85	22	
16	08	23	BSV
17	87	24	
18	18	25	CAN
19	19	26	EMV
1A	92	27	
1B	8F	28	
1C	1C	29	FSV
1D	1D	30	GSV
1E	1E	31	RSV
1F	1F	32	USV
20	80	33	
21	81	34	
22	82	35	
23	83	36	
24	84	37	
25	0A	38	LFV
26	17	39	ETB
27	1B	40	ESC
28	88	41	
29	89	42	
2A	8A	43	
2B	8B	44	
2C	8C	45	
2D	05	46	ENQ
2E	06	47	ACK
2F	07	48	BEL
30	90	49	
31	91	50	
32	16	51	SYN
33	93	52	
34	94	53	
35	95	54	
36	96	55	
37	04	56	EOT
38	98	57	
39	99	58	
3A	9A	59	
3B	9B	60	
3C	14	61	DC4
3D	15	62	NAK
3E	9E	63	
3F	1A	64	SUB
40	20	65	∇
41	A0	66	
42	A1	67	
43	A2	68	
44	A3	69	
45	A4	70	
46	A5	71	
47	A6	72	
48	A7	73	
49	A8	74	
4A	5B	75	[¢
4B	2E	76	.
4C	3C	77	<
4D	28	78	(
4E	2B	79	+
4F	21	80	! |
50	26	81	&
51	A9	82	
52	AA	83	
53	AB	84	
54	AC	85	
55	AD	86	
56	AE	87	
57	AF	88	
58	B0	89	
59	B1	90	
5A	5D	91	] !
5B	24	92	$
5C	2A	93	*
5D	29	94	)
5E	3B	95	;
5F	5E	96	^ ¬
60	2D	97	-
61	2F	98	/
62	B2	99	
63	B3	100	
64	B4	101	
65	B5	102	
66	B6	103	
67	B7	104	
68	B8	105	
69	B9	106	
6A	7C	107	¦
6B	2C	108	,
6C	25	109	%
6D	5F	110	
6E	3E	111	>
6F	3F	112	?
70	BA	113	
71	BB	114	
72	BC	115	
73	BD	116	
74	BE	117	
75	BF	118	
76	C0	119	
77	C1	120	
78	C2	121	
79	60	122	`
7A	3A	123	:
7B	23	124	#
7C	40	125	@
7D	27	126	'
7E	3D	127	=
7F	22	128	"
80	C3	129	
81	61	130	a
82	62	131	b
83	63	132	c
84	64	133	d
85	65	134	e
86	66	135	f
87	67	136	g
88	68	137	h
89	69	138	i
8A	C4	139	
8B	C5	140	
8C	C6	141	
8D	C7	142	
8E	C8	143	
8F	C9	144	
90	CA	145	
91	6A	146	j
92	6B	147	k
93	6C	148	l
94	6D	149	m
95	6E	150	n
96	6F	151	o
97	70	152	p
98	71	153	q
99	72	154	r
9A	CB	155	
9B	CC	156	
9C	CD	157	
9D	CE	158	
9E	CF	159	
9F	D0	160	
A0	D1	161	
A1	7E	162	~
A2	73	163	s
A3	74	164	t
A4	75	165	u
A5	76	166	v
A6	77	167	w
A7	78	168	x
A8	79	169	y
A9	7A	170	z
AA	D2	171	
AB	D3	172	
AC	D4	173	
AD	D5	174	
AE	D6	175	
AF	D7	176	
B0	D8	177	
B1	D9	178	
B2	DA	179	
B3	DB	180	
B4	DC	181	
B5	DD	182	
B6	DE	183	
B7	DF	184	
B8	E0	185	
B9	E1	186	
BA	E2	187	
BB	E3	188	
BC	E4	189	
BD	E5	190	
BE	E6	191	
BF	E7	192	
C0	7B	193	{
C1	41	194	A
C2	42	195	B
C3	43	196	C
C4	44	197	D
C5	45	198	E
C6	46	199	F
C7	47	200	G
C8	48	201	H
C9	49	202	I
CA	E8	203	
CB	E9	204	
CC	EA	205	
CD	EB	206	
CE	EC	207	
CF	ED	208	
D0	7D	209	}
D1	4A	210	J
D2	4B	211	K
D3	4C	212	L
D4	4D	213	M
D5	4E	214	N
D6	4F	215	O
D7	50	216	P
D8	51	217	Q
D9	52	218	R
DA	EE	219	
DB	EF	220	
DC	F0	221	
DD	F1	222	
DE	F2	223	
DF	F3	224	
E0	5C	225	\
E1	9F	226	
E2	53	227	S
E3	54	228	T
E4	55	229	U
E5	56	230	V
E6	57	231	W
E7	58	232	X
E8	59	233	Y
E9	5A	234	Z
EA	F4	235	
EB	F5	236	
EC	F6	237	
ED	F7	238	
EE	F8	239	
EF	F9	240	
F0	30	241	0
F1	31	242	1
F2	32	243	2
F3	33	244	3
F4	34	245	4
F5	35	246	5
F6	36	247	6
F7	37	248	7
F8	38	249	8
F9	39	250	9
FA	FA	251	
FB	FB	252	
FC	FC	253	
FD	FD	254	
FE	FE	255	
FF	FF	256	

MCS DATA PROCESSING

On output, the characteristics of the line procedure are important, since certain control codes and, in some cases, graphic symbols, are not applicable, see the tables of individual line procedures under "Output Normal Mode".

For this reason the table "MCS Data Processing" is based on the BSC2780/VIP line procedure. The principles that govern the handling of data formats on output for the BSC2780/VIP line procedure, must be applied accordingly to other line procedures.

The EBCDIC values chosen for illustrating MCS data processing are,

- "0C" corresponding to the control code FF▽ which is valid for all line procedures
- "32" corresponding to the control code SYN which does not apply in output mode for the BSC2780/VIP line procedure
- "C7" corresponding to the letter G which is a typical example of a displayable graphic symbol
- "FD" having no equivalent either as a control code or a graphic symbol in any line procedure.

The points to be noted in the table, are

- Control codes in mark form, which are applicable to the line procedure, are translated as 1-byte hexadecimal values on output
- Control codes in mark form, which are not applicable to the line procedure, pass unchanged on output, and are deleted if preceded by "repeat"
- Character-encoded hexadecimal values are processed in exactly the same manner, irrespective of the fact whether these values correspond to control codes, graphic symbols or to neither
- 1-byte hexadecimal values on input are handled accordingly as follows,
  - if representing control codes, are processed in exactly the same manner, with the notable exceptions of HT▽ and ESC which in normal mode pass unchanged as "05" and "27" respectively
  - otherwise, they are processed regardless as graphic symbols
- 1-byte hexadecimal values on output are handled differently in normal mode, where no "repeat" precedes the value, namely,
  - if representing control codes, applicable to the line procedure, or if corresponding to a displayable graphic symbol, are passed unchanged
  - otherwise, they are translated into character-encoded hexadecimal values.

The 1-byte hexadecimal value is generated by the activation by a touch-key, represented either as a control code or graphic symbol on the keyboard. For this reason the value "FD" shown in the table in input mode is only meaningful when considered as a non-standard code, for example, for national keyboard options.

VIP headers and trailers are dealt with in detail under each of the process modes.

MCS Data Processing (based on BSC2780/VIP line procedure)					
Data from Terminal or in Buffer	INPUT			OUTPUT	
	Normal	Marked	Unedited	Normal	Unedited
><FF▽ ><RO2><FF▽	><FF▽ delete	><FF▽ delete	><FF▽ ><RO2><FF▽	"OC" "OCOC"	><FF▽ ><RO2><FF▽
><SYN ><RO2><SYN	><SYN delete	><SYN delete	><SYN ><RO2><SYN	><SYN delete	><SYN ><RO2><SYN
><COC ><RO2><COC	"OC" "OCOC"	"OC" "OCOC"	><COC ><RO2><COC	"OC" "OCOC"	><COC ><RO2><COC
><C32 ><RO2><C32	"32" "3232"	"32" "3232"	><C32 ><RO2><C32	"32" "3232"	><C32 ><RO2><C32
><CC7 ><RO2><CC7	"C7" "C7C7"	"C7" "C7C7"	><CC7 ><RO2><CC7	"C7" "C7C7"	><CC7 ><RO2><CC7
><CFD ><RO2><CFD	"FD" "FDFD"	"FD" "FDFD"	><CFD ><RO2><CFD	"FD" "FDFD"	><CFD ><RO2><CFD
"OC" ><RO2"OC"	delete not HT▽/ESC "OCOC"	><FF▽ "OCOC"	"OC" ><RO2"OC"	"OC" "OCOC"	"OC" ><RO2"OC"
"32" ><RO2"32"	delete "3232"	><SYN "3232"	"32" ><RO2"32"	><C32 "3232"	"32" ><RO2"32"
"C7" ><RO2"C7"	"C7" "C7C7"	"C7" "C7C7"	"C7" ><RO2"C7"	"C7" "C7C7"	"C7" ><RO2"C7"
"FD" ><RO2"FD"	"FD" "FDFD"	"FD" "FDFD"	"FD" ><RO2"FD"	><CFD "FDFD"	"FD" ><RO2"FD"

Processing on Input

The 3 transmission modes for processing on input are,

- input marked mode
- input normal mode
- input unedited mode.

The table "Input Marked Mode" is applicable for all line procedures, and gives the complete list of control codes known to the "stream processor" in their mark form.

These control codes in mark form, when appearing singly, pass unchanged to the user buffer in all the 3 input modes.

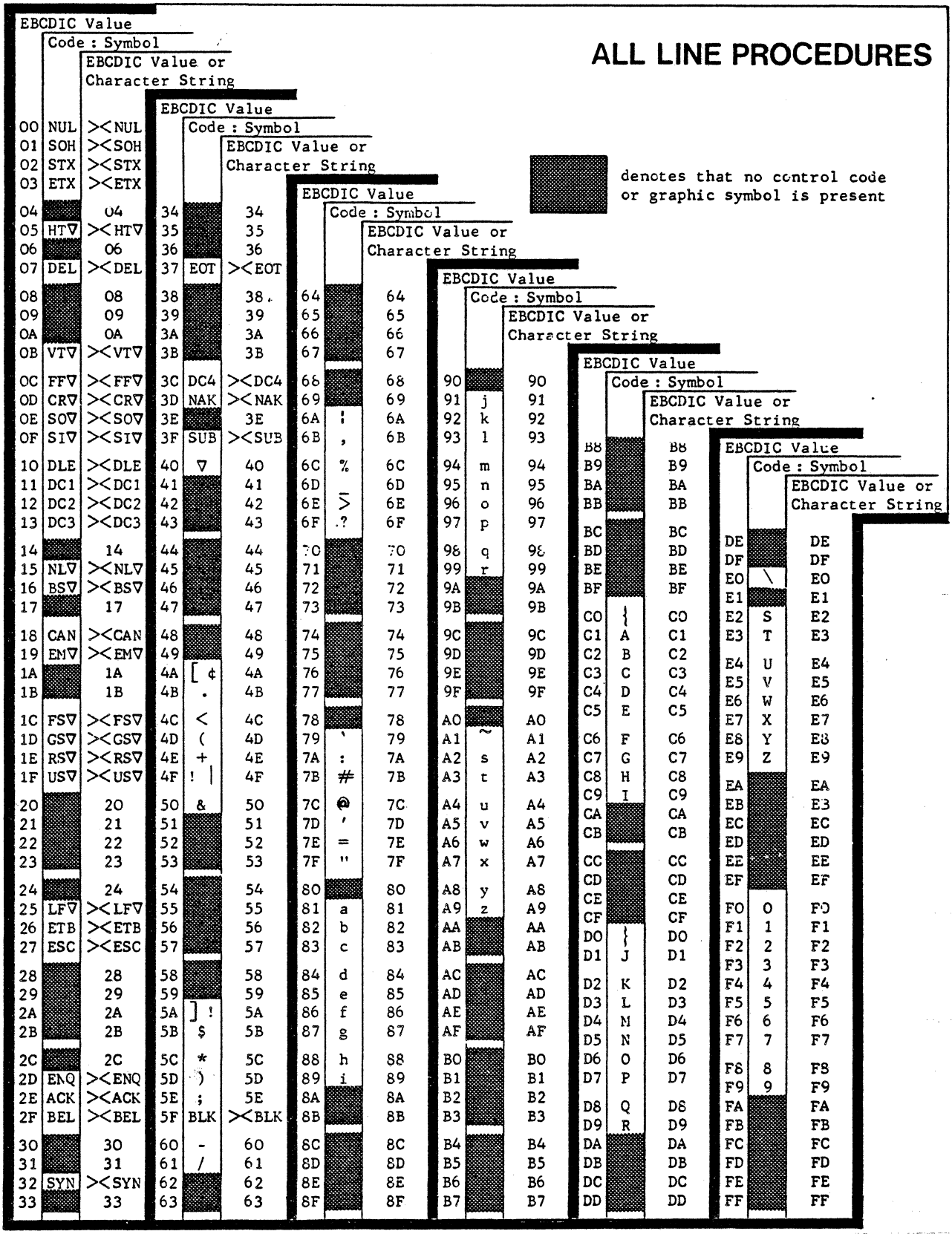
The 2 control code exceptions in input normal mode are HTV and ESC, which unlike the other control codes, are not deleted from the user buffer.

All other values, which are not control codes, are treated as graphic symbols, whether or not a displayable symbol exists, and pass unchanged, when appearing singly, to the user buffer.

The input unedited mode passes all data unprocessed to the user buffer, and all verification of data must be performed by the MCS application.

INPUT MARKED MODE

ALL LINE PROCEDURES



Input Marked Mode

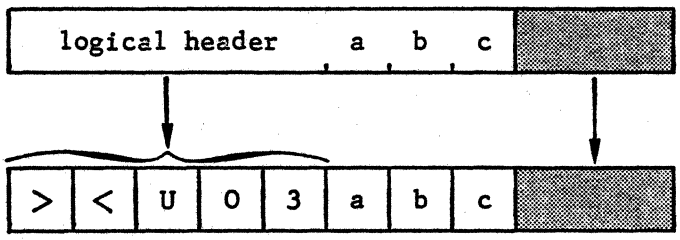
Mode Entry

The input marked mode is entered

- at network generation by the QUEUE command pertaining to the terminal queue specifying the parameter IM=MK
- during the communications session to override whatever input mode has been declared at network generation
  - either by the network control command MTE specifying the terminal and the IMARK parameter
  - or by the terminal operator command $*$MTE specifying IMARK.

Treatment of Data

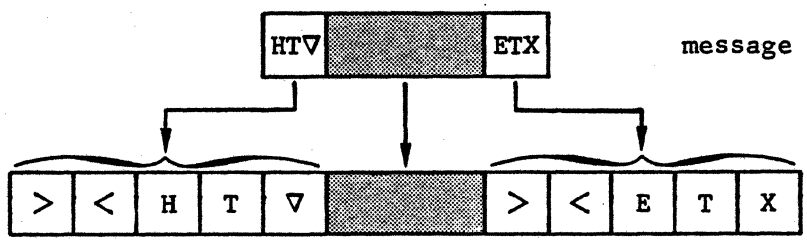
- VIP-headers are translated into mark form and passed to the user buffer.



header generated by terminal in VIP line procedure

data transmitted to buffer

- Control codes and trailers are translated into mark form and passed to the user buffer.

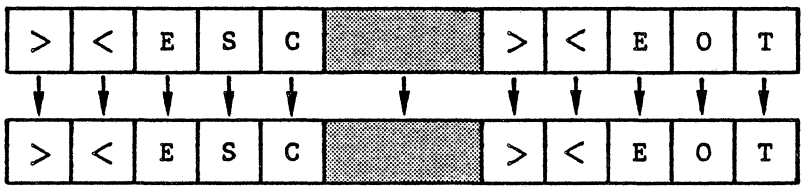


message keyed in on terminal

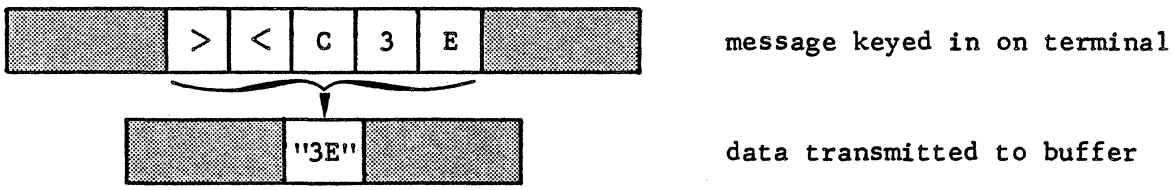
data transmitted to user buffer

Input Marked Mode
Treatment of Data
(continued)

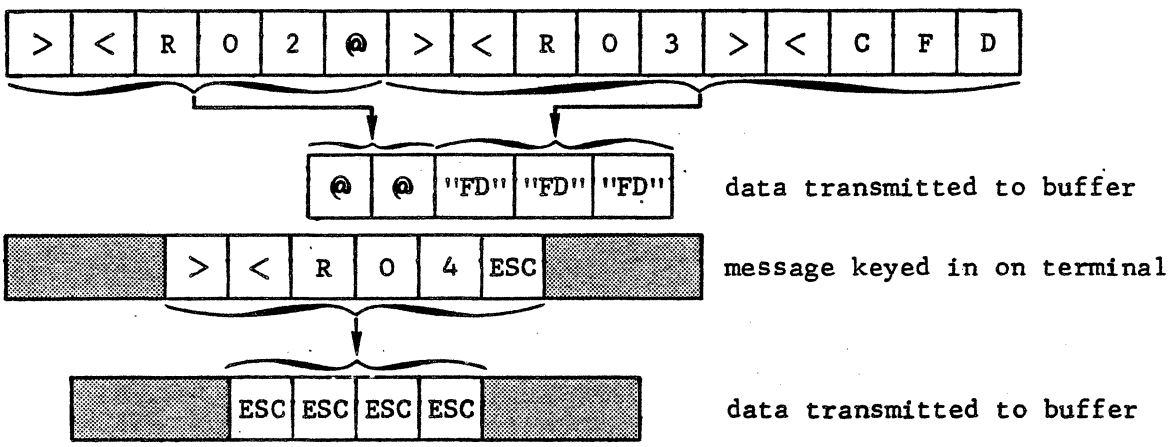
- Control codes and trailers in mark form are passed unchanged to the user buffer.



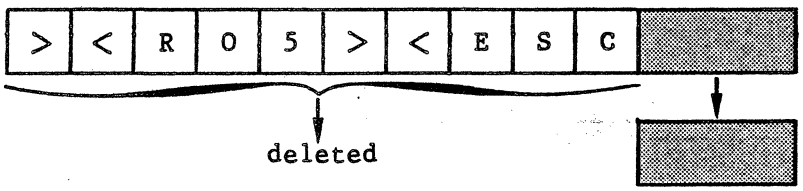
- The character-encoding mark >< C passes the following 2 hexadecimal-digits as a 1-byte hexadecimal value to the user buffer.



- The repeat mark >< R repeats the graphic symbol, character-encoded hexadecimal value or control code, as a 1-byte hexadecimal value.



- The sequence of the repeat mark followed by a control code in mark form is deleted from the user buffer.



- All other character strings are passed unchanged to the buffer, notably,
  - >< Cxy, where xy is not a hexadecimal value
  - >< Rab, where ab is not a decimal value.

Input Normal Mode

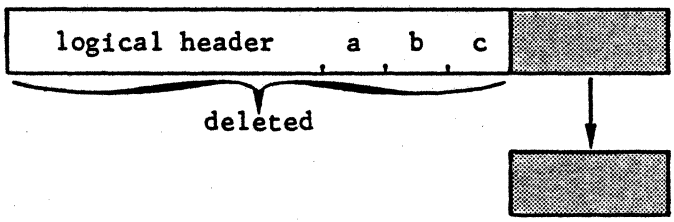
Mode Entry

The input normal mode is entered

- at network generation by the QUEUE command pertaining to the terminal queue
  - either not specifying the IM parameter, that is, "normal" is the default input mode
  - or specifying the parameter IM=NL
- during the communications session to override whatever input mode has been declared at network generation
  - either by the network control command MTE specifying the terminal and the INORM parameter
  - or by the terminal operator command $*$MTE specifying INORM.

Treatment of Data

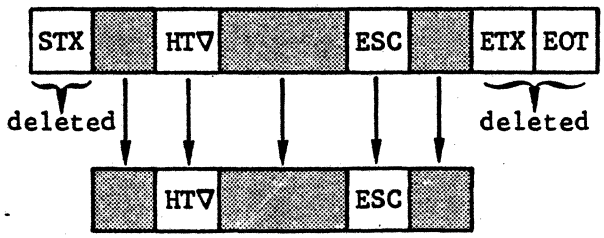
- VIP-headers are deleted from the user buffer.



header generated by terminal in VIP line procedure

data transmitted to buffer

- Except for HT∇ and ESC, all control codes are deleted from the buffer.

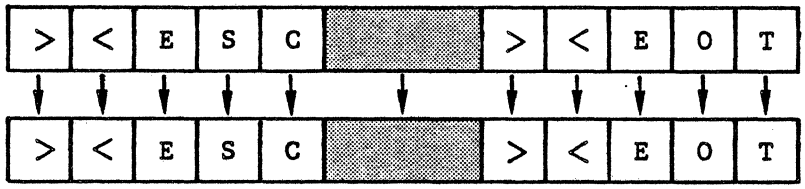


message keyed in on terminal

data transmitted to buffer

Input Normal Mode
Treatment of Data
(continued)

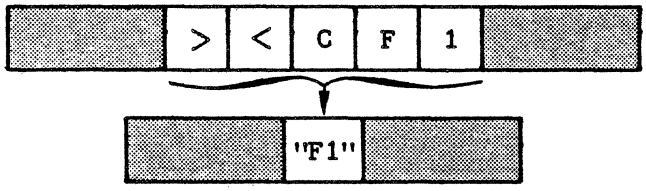
- Control codes and trailers in mark form are passed unchanged to the user buffer.



message keyed in on terminal

data transmitted to user buffer

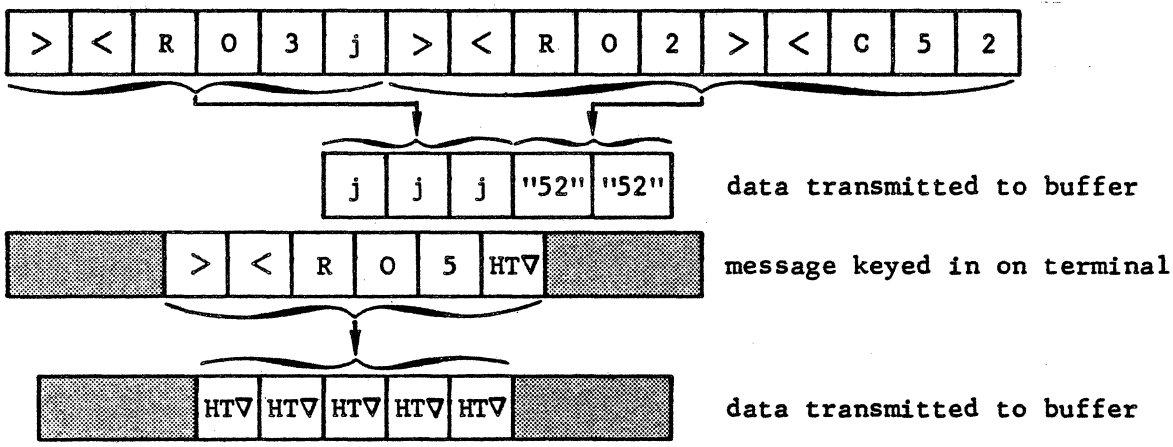
- The character-encoding mark >< C passes the following 2 hexadecimal-digits as a 1-byte hexadecimal value to the user buffer.



message keyed in on terminal

data transmitted to buffer

- The repeat mark >< R repeats the graphic symbol, character-encoded hexadecimal value or control code, as a 1-byte hexadecimal value.

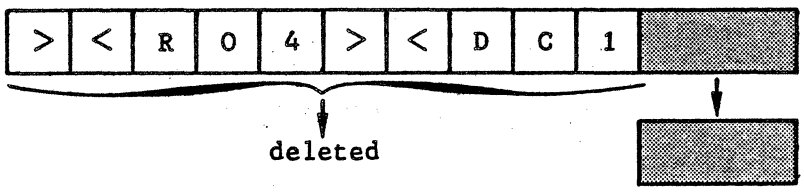


data transmitted to buffer

message keyed in on terminal

data transmitted to buffer

- The sequence of the repeat mark followed by a control code in mark form is deleted from the user buffer.



message keyed in on terminal

data transmitted to user buffer

- All other character strings are passed unchanged to the buffer, notably,
  - >< Cxy, where xy is not a hexadecimal value
  - >< Rab, where ab is not a decimal value.

Input Unedited Mode

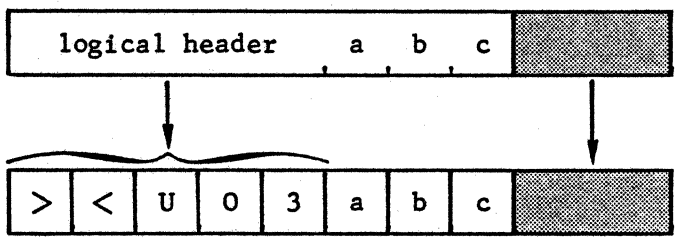
Mode Entry

The input unedited mode is entered

- at network generation by the QUEUE command pertaining to the terminal queue specifying the parameter IM=UN
- during the communications session to override whatever input mode has been declared at network generation
  - either by the network control command MTE specifying the terminal and the INEDT parameter
  - or by the terminal operator command $*$MTE specifying INEDT.

Treatment of Data

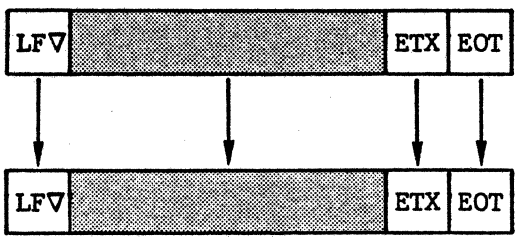
- VIP-headers are translated into mark form and passed to the user buffer.



header generated by terminal in VIP line procedure

data transmitted to buffer

- Control codes and trailers are passed unchanged to the user buffer.

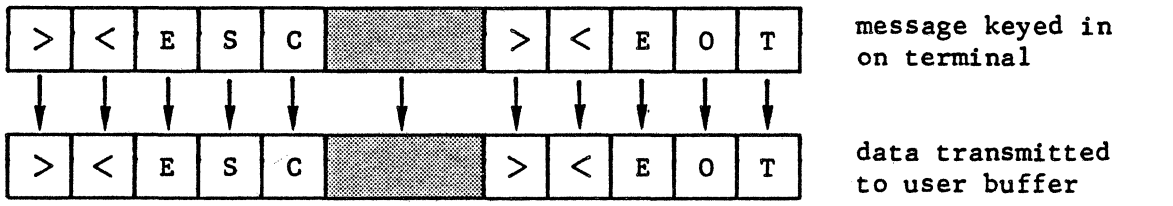


message keyed in on terminal

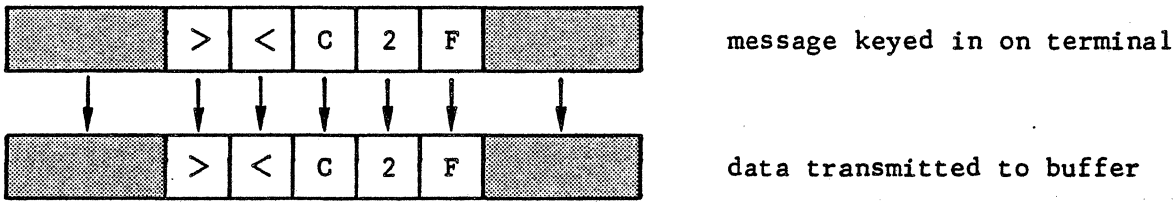
data transmitted to buffer

Input Unedited Mode
Treatment of Data
(continued)

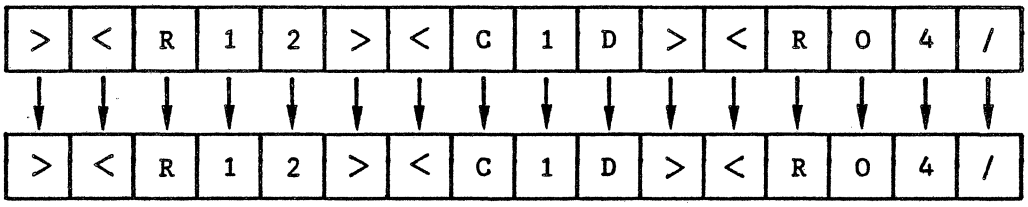
- Control codes and trailers in mark form are passed unchanged to the user buffer.



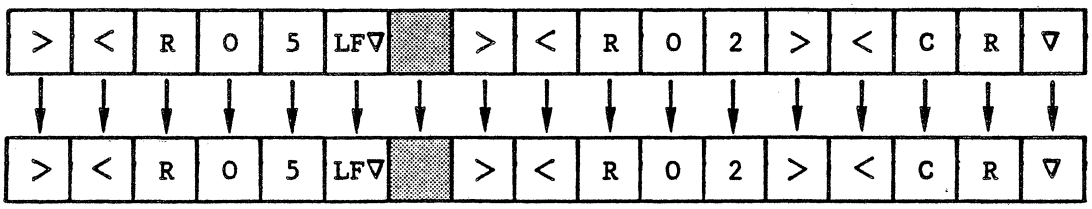
- The character-encoding mark ><C, even if followed by 2 hexadecimal-digits, does not perform the "character-encoding" function and is passed unchanged to the user buffer.



- The repeat mark ><R, even if followed by 2 decimal-digits, does not perform the "repeat" function and is passed unchanged to the user buffer.



- The sequence of the repeat mark followed immediately by a control code, either as a 1-byte hexadecimal value or in mark form, is passed unchanged to the user buffer.



- All data is treated as character strings and passed unchanged to the user buffer.

Processing on Output

The 2 transmission modes for processing on output are,

- output normal mode
- output unedited mode.

For output normal mode, tables are given for each line procedure, since the way in which certain control codes are handled, depends on whether they are applicable to the line procedure.

In addition, certain graphic symbols are treated as non-standard, since the receiving terminal, like the IBM3270, cannot display them.

For the IBM3270, the following graphic symbols are not displayed,

- 79= ` , open single quote
- A1= ~ , tilde
- C0= { , open brace
- D0= } , close brace
- E0= \ , back-slash.

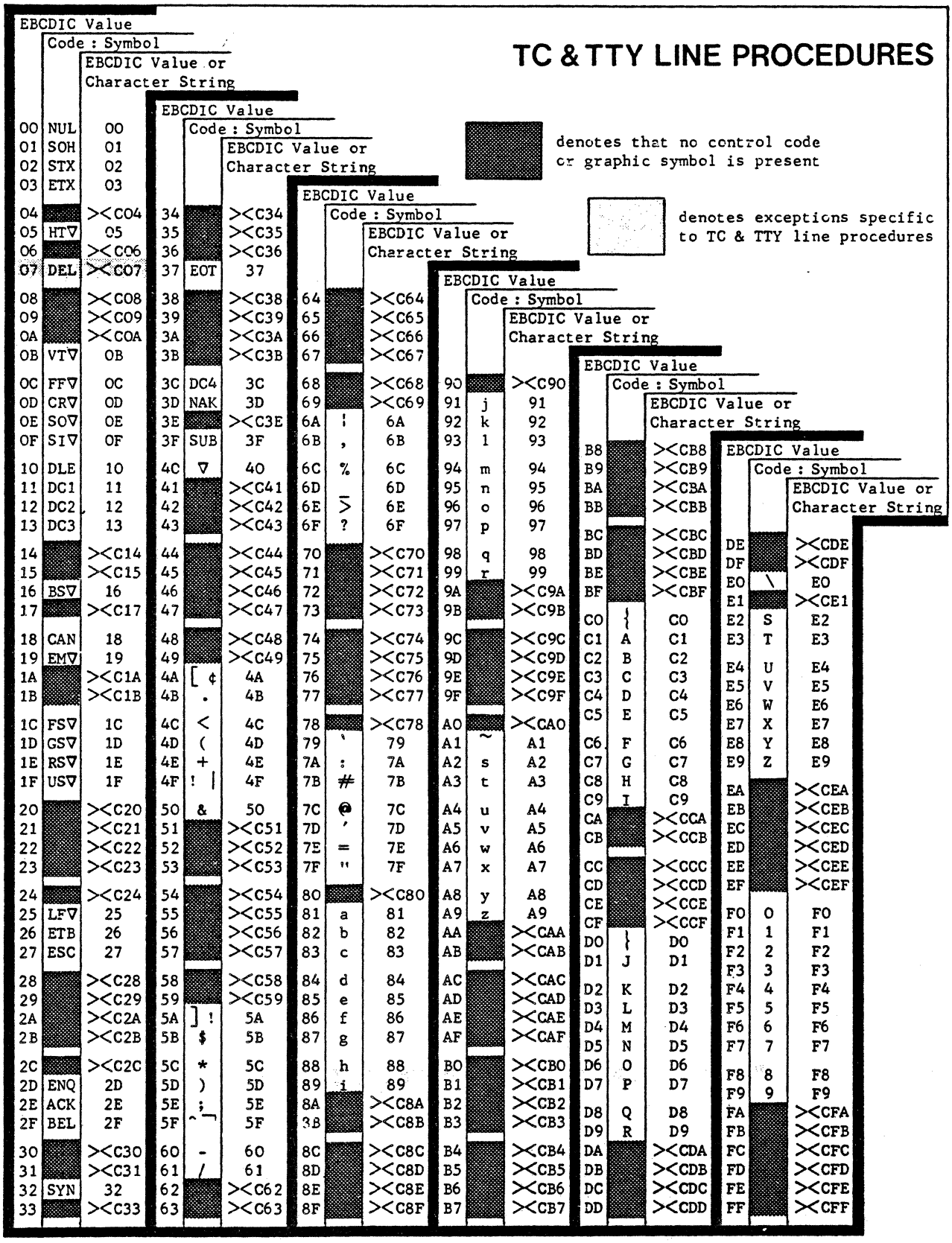
In addition, 7D displays as an apostrophe (') and not as a close single quote (').

In each of the output normal mode tables, control codes that are not applicable for output for the line procedure, are shown in character-encoded form.

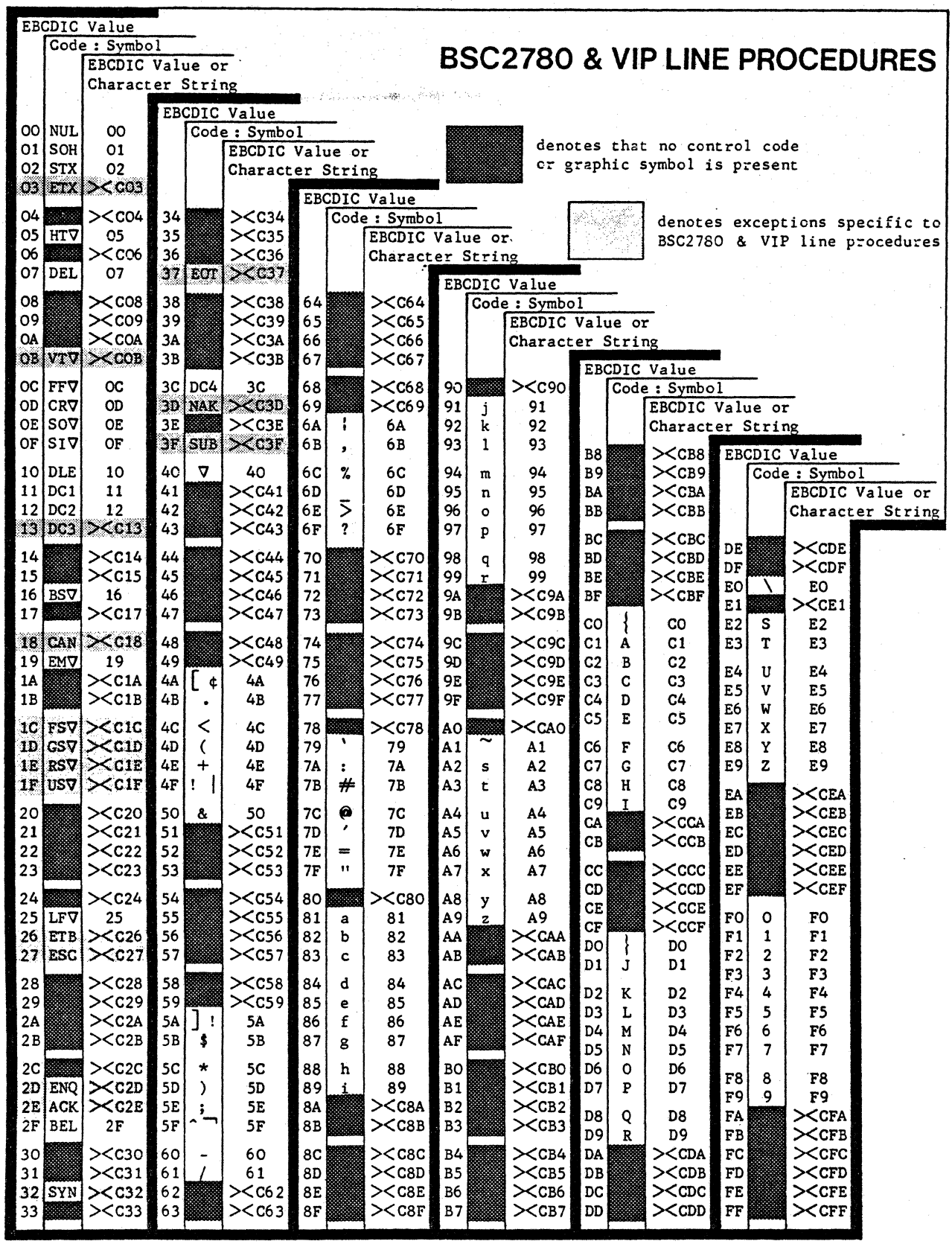
In this respect, non-applicable control codes are treated in the same way as transmissions for which no graphic symbol exists.

OUTPUT NORMAL MODE

TC & TTY LINE PROCEDURES

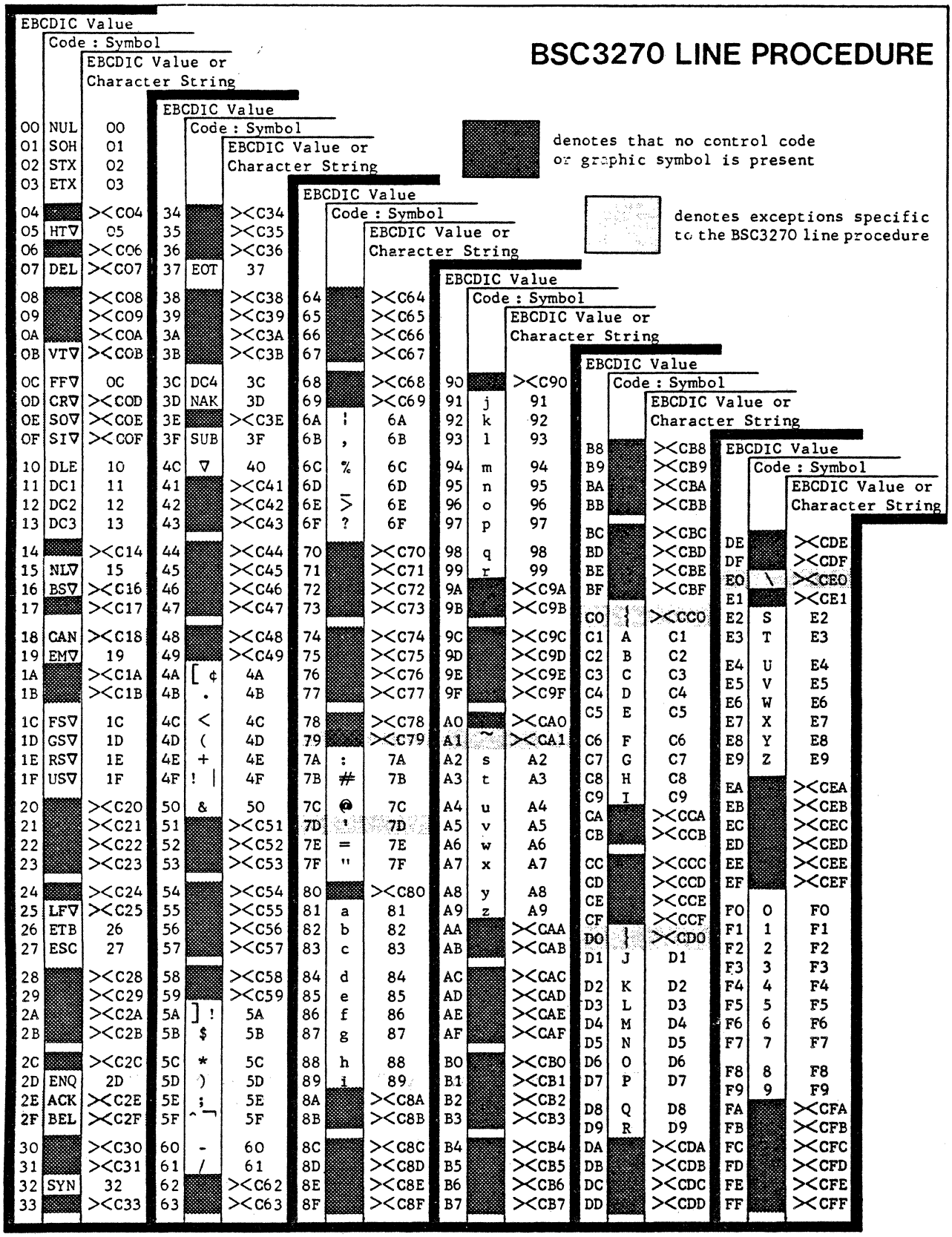


OUTPUT NORMAL MODE



OUTPUT NORMAL MODE

BSC3270 LINE PROCEDURE



EBCDIC Value	Code : Symbol	EBCDIC Value or Character String
00	NUL	00
01	SOH	01
02	STX	02
03	ETX	03
04		><C04
05	HT∇	05
06		><C06
07	DEL	><C07
08		><C08
09		><C09
0A		><C0A
0B	VT∇	><C0B
0C	FF∇	0C
0D	CR∇	><C0D
0E	SO∇	><C0E
0F	SI∇	><C0F
10	DLE	10
11	DC1	11
12	DC2	12
13	DC3	13
14		><C14
15	NL∇	15
16	BS∇	><C16
17		><C17
18	CAN	><C18
19	EM∇	19
1A		><C1A
1B		><C1B
1C	FS∇	1C
1D	GS∇	1D
1E	RS∇	1E
1F	US∇	1F
20		><C20
21		><C21
22		><C22
23		><C23
24		><C24
25	LF∇	><C25
26	ETB	26
27	ESC	27
28		><C28
29		><C29
2A		><C2A
2B		><C2B
2C		><C2C
2D	ENQ	2D
2E	ACK	><C2E
2F	BEL	><C2F
30		><C30
31		><C31
32	SYN	32
33		><C33
34		><C34
35		><C35
36		><C36
37	EOT	37
38		><C38
39		><C39
3A		><C3A
3B		><C3B
3C	DC4	3C
3D	NAK	3D
3E		><C3E
3F	SUB	3F
40	∇	40
41		><C41
42		><C42
43		><C43
44		><C44
45		><C45
46		><C46
47		><C47
48		><C48
49		><C49
4A	[¢	4A
4B	.	4B
4C	<	4C
4D	(	4D
4E	+	4E
4F	! |	4F
50	&	50
51		><C51
52		><C52
53		><C53
54		><C54
55		><C55
56		><C56
57		><C57
58		><C58
59		><C59
5A	] !	5A
5B	$	5B
5C	*	5C
5D	)	5D
5E	;	5E
5F	^ ¬	5F
60	-	60
61	/	61
62		><C62
63		><C63
64		><C64
65		><C65
66		><C66
67		><C67
68		><C68
69		><C69
6A	¦	6A
6B	,	6B
6C	%	6C
6D	_	6D
6E	>	6E
6F	?	6F
70		><C70
71		><C71
72		><C72
73		><C73
74		><C74
75		><C75
76		><C76
77		><C77
78		><C78
79		><C79
7A	:	7A
7B	#	7B
7C	@	7C
7D	'	7D
7E	=	7E
7F	"	7F
80		><C80
81	a	81
82	b	82
83	c	83
84	d	84
85	e	85
86	f	86
87	g	87
88	h	88
89	i	89
8A		><C8A
8B		><C8B
8C		><C8C
8D		><C8D
8E		><C8E
8F		><C8F
90		><C90
91	j	91
92	k	92
93	l	93
94	m	94
95	n	95
96	o	96
97	p	97
98	q	98
99	r	99
9A		><C9A
9B		><C9B
9C		><C9C
9D		><C9D
9E		><C9E
9F		><C9F
A0		><CA0
A1	~	><CA1
A2	s	A2
A3	t	A3
A4	u	A4
A5	v	A5
A6	w	A6
A7	x	A7
A8	y	A8
A9	z	A9
AA		><CAA
AB		><CAB
AC		><CAC
AD		><CAD
AE		><CAE
AF		><CAF
B0		><CB0
B1		><CB1
B2		><CB2
B3		><CB3
B4		><CB4
B5		><CB5
B6		><CB6
B7		><CB7
B8		><CB8
B9		><CB9
BA		><CBA
BB		><CBB
BC		><CBC
BD		><CBD
BE		><CBE
BF		><CBF
C0	{	><CC0
C1	A	C1
C2	B	C2
C3	C	C3
C4	D	C4
C5	E	C5
C6	F	C6
C7	G	C7
C8	H	C8
C9	I	C9
CA		><CCA
CB		><CCB
CC		><CCC
CD		><CCD
CE		><CCE
CF		><CCF
D0	}	><CD0
D1	J	D1
D2	K	D2
D3	L	D3
D4	M	D4
D5	N	D5
D6	O	D6
D7	P	D7
D8	Q	D8
D9	R	D9
DA		><CDA
DB		><CDB
DC		><CDC
DD		><CDD
DE		><CDE
DF		><CDF
E0	\	><CE0
E1		><CE1
E2	S	E2
E3	T	E3
E4	U	E4
E5	V	E5
E6	W	E6
E7	X	E7
E8	Y	E8
E9	Z	E9
EA		><CEA
EB		><CEB
EC		><CEC
ED		><CED
EE		><CEE
EF		><CEF
F0	0	F0
F1	1	F1
F2	2	F2
F3	3	F3
F4	4	F4
F5	5	F5
F6	6	F6
F7	7	F7
F8	8	F8
F9	9	F9
FA		><CFA
FB		><CFB
FC		><CFC
FD		><CFD
FE		><CFE
FF		><CFF

Output Normal Mode

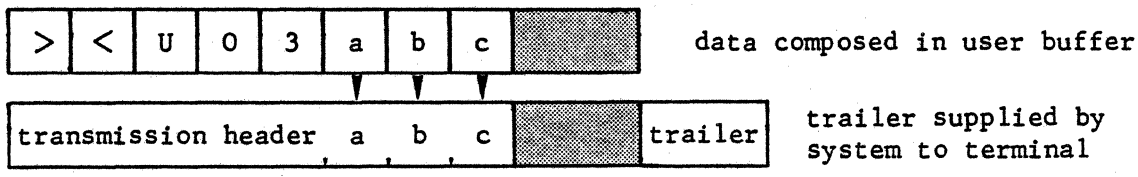
Mode Entry

The output normal mode is entered

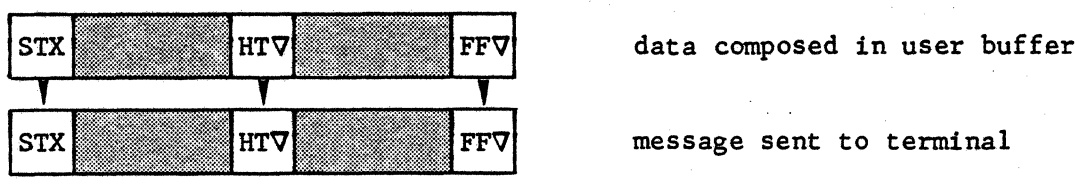
- at network generation by the QUEUE command pertaining to the terminal queue
  - either not specifying the OM parameter, that is, "normal" is the default output mode
  - or specifying the parameter OM=NL
- during the communications session to override the "unedited" mode declared at network generation
  - either by the network control command MTE specifying the terminal and the ONORM parameter
  - or by the terminal operator command $*$MTE specifying ONORM.

Treatment of Data

- VIP-headers can be provided in mark form to be passed to the terminal. The system provides trailers where none are specified by the user.

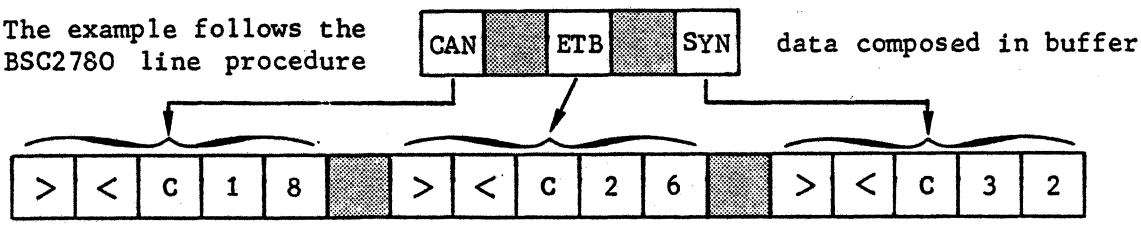


- Control codes, applicable to the line procedure for output, are passed unchanged to the terminal.



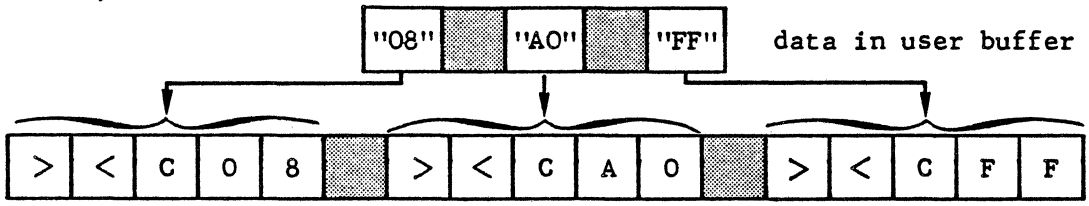
- Control codes, not applicable for output, are translated into character-encoded form and sent to the terminal.

The example follows the BSC2780 line procedure

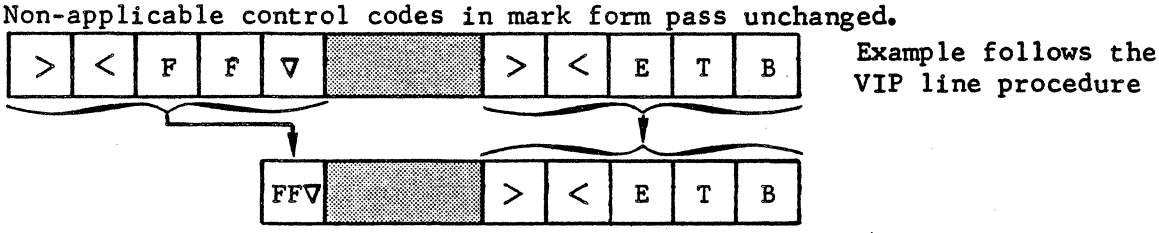


Output Normal Mode
Treatment of Data
(continued)

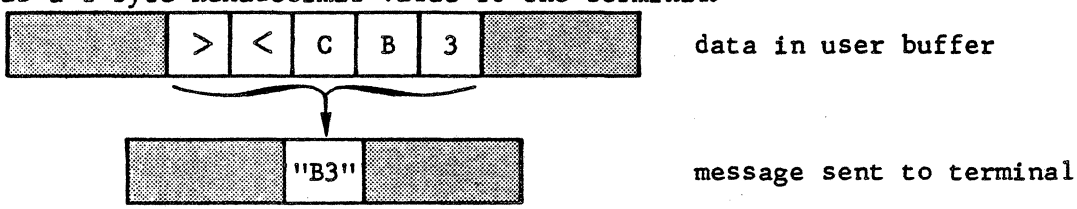
- 1-byte hexadecimal values, for which no graphic symbol or control code exists, are translated into character-encoded form.



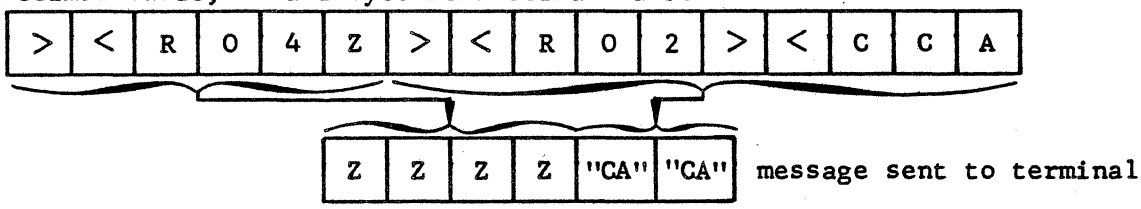
- Applicable control codes in mark form translate as 1-byte hexadecimals. Non-applicable control codes in mark form pass unchanged.



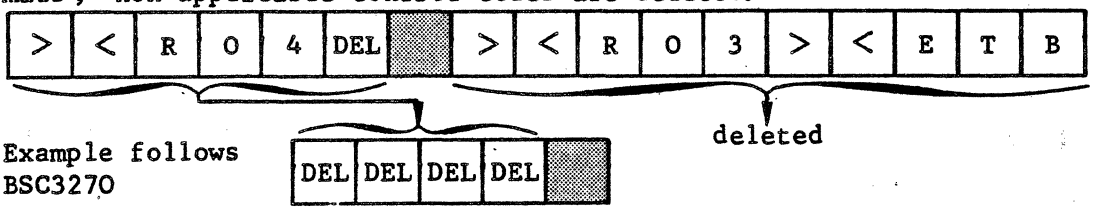
- The character-encoding mark >< C passes the following 2 hexadecimal-digits as a 1-byte hexadecimal value to the terminal.



- The repeat mark >< R repeats the graphic symbol or character-encoded hexadecimal value, as a 1-byte hexadecimal value.



- The repeat mark >< R repeats applicable control codes as 1-byte hexadecimals; non-applicable control codes are deleted.



- All other character strings are passed unchanged to the terminal, notably,
  - >< Cxy, where xy is not a hexadecimal value
  - >< Rab, where ab is not a decimal value.

Output Unedited Mode

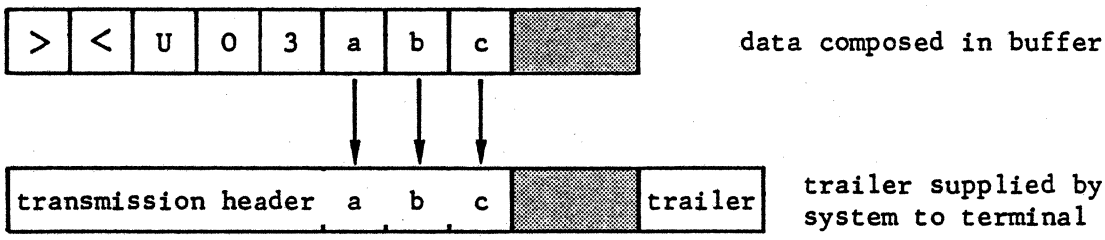
Mode Entry

The output unedited mode is entered

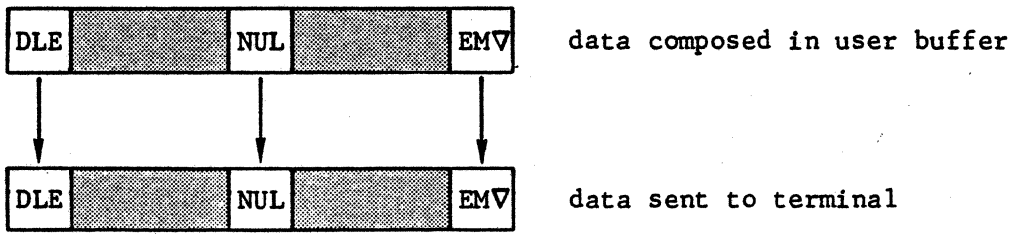
- at network generation by the QUEUE command pertaining to the terminal queue specifying the parameter OM=UN
- during the communications session to override the "normal" mode declared at network generation
  - either by the network control command MTE specifying the terminal and the ONEDT parameter
  - or by the terminal operator command $*$MTE specifying ONEDT.

Treatment of Data

- VIP-headers can be provided in mark form to be passed to the terminal. The system provides trailers where none are specified by the user.

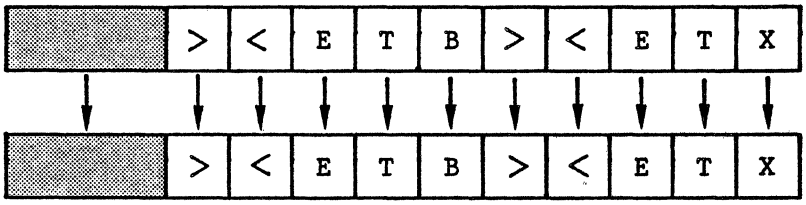


- Control codes are passed unchanged to the terminal.

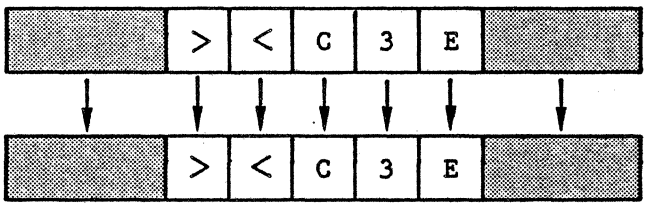


Output Unedited Mode
Treatment of Data
(continued)

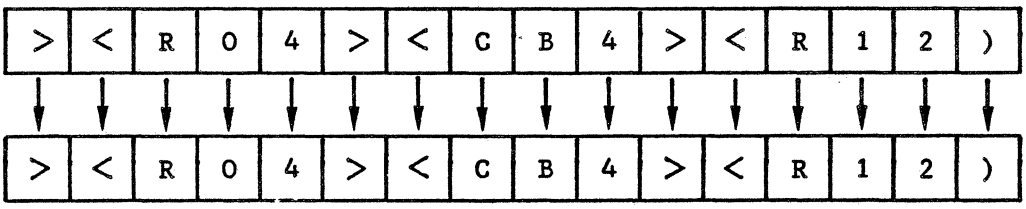
- Control codes and trailers in mark form are passed unchanged to the terminal.



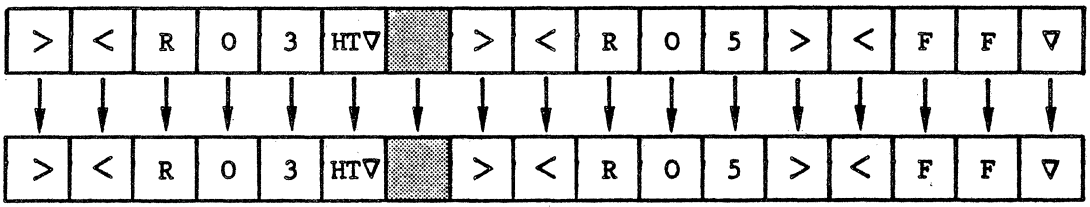
- The character-encoding mark ><C, even if followed by 2 hexadecimal-digits, does not perform the "character-encoding" function and is passed unchanged to the terminal.



- The repeat mark ><R, even if followed by 2 decimal-digits, does not perform the "repeat" function and is passed unchanged to the terminal.



- The sequence of the repeat mark followed immediately by a control code, either as a 1-byte hexadecimal value or in mark form, is passed unchanged to the terminal.



- All data is treated as character strings and passed unchanged to the terminal.

DATA REPRESENTATION

GCOS offers the programmer several ways of representing data depending on such factors as,

- the type of terminal used, which may involve a special character set
- the complexity of control functions to be activated
- the provisions to be made for the transition of the MCS application to TDS with the minimum of modifications.

Graphic Symbols

In the 2 examples "Data Representation Using Graphic Symbols", the data cited is valid for any transmission code.

Variations in representing graphic symbols are caused by,

- type of device
- national language options
- special characters.

TYPE OF DEVICE

The character set available to the user, depends on the type of the device that is used for data entry, for example,

- if the data is entered in the form of cards, the only character set possible is standard EBCDIC, or part of that set allowed on the keypunch, since lower case letters are not represented
- if, however, the data is entered under IOF, from a terminal having upper and lower case capability, that is, having the "shift-in/shift-out" function, the character set is greatly extended.

Conversely, characters are displayed according to the type of the device used to display them, namely, the line printer will only display upper case letters, even if the data is entered in lower case letters from, say, a VIP7760 terminal.

NATIONAL LANGUAGE OPTIONS

The different graphic symbols used for national language options come within the category of special characters, with the one basic difference, namely,

- special characters are used as conventional symbols, primarily,
  - to delimit text, such as punctuation signs
  - to qualify text, such as monetary signs to indicate currency
- national language options, however, are symbols used in the context of readable text in a particular language, for example, Ø for Danish.

The user, in order to ensure correct processing of national language options, must specify their equivalents in his application, see "Special Characters" and "Numeric Values".

SPECIAL CHARACTERS

The DPS 7 CPU internal code is EBCDIC, the graphic representation of which is used by local unit record devices, such as, the system console, the card reader and/or punch, and the line printer.

Most terminals, however, use the ISO ASCII code in which the graphic representation of special characters differs in some cases from that of EBCDIC.

This means that the user, in order to display certain special characters on his terminal, must specify their equivalents* in his application.

In the ASCII and EBCDIC tables, where 2 graphic symbols appear against the code, the graphic symbol on the right is the EBCDIC representation.

Example of Using
Special Characters

- Example of special characters rendered differently :

EBCDIC graphic, used by GCOS internally :	¢	|	!	¬
ISO ASCII graphic, used by VIP terminals :	[	!	]	^

- Text to be displayed on VIP terminal :

```
VERSION DATED : [ 80/03/29 ]
```

- What the user must declare in the MCS application:

MCS COBOL

```
DATA DIVISION.
  77 VERS PIC X(28)  VALUE "VERSION DATED : ¢ 80/03/29 !" .
```

GPL

```
DCL VERS CHAR(28) INIT ("VERSION DATED : ¢ 80/03/29 !");
```

* The term "equivalent" need not necessarily mean the graphic symbol equivalent.

Numeric values of the special character or national language option are used in such cases where there are no graphic symbol equivalents

- either on the device for data entry or display
- or in the standard character set recognized by GCOS.

Data Representation
Using
Graphic Symbols

To send the text `* HERE IS DATA-ENTRY (80/03/29 VERSION) *`

MCS COBOL

```
DATA DIVISION.
  77 START PIC X(41) VALUE "* HERE IS DATA-ENTRY
                            (80/03/29 VERSION) *".

COMMUNICATION SECTION.
  CD CD-OUT OUTPUT
    only user-initialized CD-output parameters are shown
    DESTINATION COUNT  COUNT-OUT
    TEXT LENGTH        LENGTH-OUT
    DESTINATION        DESTINATION-OUT.

PROCEDURE DIVISION.
  MOVE 1 TO COUNT-OUT.
  MOVE 41 TO LENGTH-OUT.
  MOVE destination TO DESTINATION-OUT.
  SEND CD-OUT FROM START WITH EMI.
```

GPL

```
$H_CD OUTPUT, PREFIX='USER_';

only user-initialized CD-output parameters are shown

USER_DESTINATION_COUNT=1;

USER_TEXT_LENGTH=41;

USER_QUEUE_NAME="destination";
 :
 :
DCL START CHAR(41)
    INIT ("* HERE IS DATA-ENTRY (80/03/29 VERSION) *");

$H_SEND 'ADDR(USER_OUTPUT_CD)', INADDR='ADDR(START)',
        ENDCHAR=EMI;
```

Data Representation
Using
Graphic Symbols

To receive the message **STOP** requesting "end-of-application"

MCS COBOL

```
DATA DIVISION.
  01 INBUF.
    02 INB1 PIC X(2000).
    02 INB2 REDEFINES INB1.
      03 INB21 PIC X(4).
      03 INB22 PIC X(1996).

COMMUNICATION SECTION.
  CD CD-IN INPUT
    only user-initialized CD-input parameters are shown
    TEXT LENGTH     LENGTH-IN

PROCEDURE DIVISION.
  MOVE 2000 TO LENGTH-IN.
  RECEIVE CD-IN MESSAGE INTO INBUF.
  IF INB21="STOP" GO TO TERM-PROG.
   .
   .
   .
  TERM-PROG.
   .
   .
   .
```

GPL

```
$H_CD INPUT PREFIX='USER_';

DCL INBUF CHAR(2000);

$H_RECEIVE 'ADDR(USER_INPUT_CD)' , OUTADDR='ADDR(INBUF)',
           LENGTH=2000;

IF SUBSTR(INBUF,1,4)="STOP" THEN GO TO TERM-PROG;
 .
 .
 .
TERM-PROG:;
 .
 .
 .
```

Representing Graphic Symbols

The 3 ways of representing graphic symbols are,

- direct use of graphic symbols
- numeric values
- mark form.

DIRECT USE OF GRAPHIC SYMBOLS

Without exception, all numeric characters and alphabetic capitals are entered directly, since these are valid for any transmission code, and for any keying-in and receiving devices.

The previous 2 examples "Data Representation Using Graphic Symbols" illustrate the direct use of graphic symbols, both on output as well as on input to the application.

NUMERIC VALUES

Representing the graphic symbol by its numeric value enables the programmer to define any code, whether displayable or not.

By this means, special characters and national language options can be specified even at installations where these are not available.

Application development, therefore, is not restricted in any way.

The numeric value is specified according to what is expected by the compiler, namely,

- for the MCS COBOL compiler, the decimal value of the COBOL collating sequence
- for the GPL compiler, the hexadecimal EBCDIC value.

National Language Option Using Numeric Value
• To represent the character Ø : 1. Refer to the appropriate terminal manual giving the QWERTY layout for the national keyboard options of Denmark or Norway. 2. The ASCII value of Ø is 5C. 3. The standard graphic symbol for ASCII 5C is \. 4. The numeric value for ASCII 5C is, . 225, being the COBOL collating sequence value in decimal . EO, being the hexadecimal EBCDIC value. • Declare the appropriate numeric value in the MCS COBOL or GPL application respectively, if the standard graphic symbol \ is not available.

Data Representation
Using
Numeric Values

To send text **Date** containing upper and lower case letters

MCS COBOL

1. Refer to the EBCDIC table for the COBOL collating sequence values for the graphic symbols required.

```
D = 197        a = 130        t = 164        e = 134
```

2. Declare in the DATA DIVISION either form of coding :

```
77  DAT  PIC  X( 4 )  VALUE  "D"130, 164, 134"".
```

```
77  DAT  PIC  X( 4 )  VALUE  ""197, 130, 164, 134"".
```

GPL

1. Refer to the EBCDIC table for the EBCDIC values of the graphic symbols.

```
D = C4        a = 81        t = A3        e = 85
```

2. Code the constant character-string in EBCDIC values between double-quotes followed by the letter H :

```
DCL  AB  CHAR(4) ;
AB = "C481A385"H ;
```

MARK FORM

Data in mark form is a general facility of MCS, by which any type of data can be represented in an easy-to-use symbolic form, see "Symbolic Representation" at the start of the section.

For graphic symbols, the 2 types of mark form dealt with are,

- ><C, denoting character-encoding
- ><R, denoting the "repeat" function.

The choice of entering graphic symbols either in their character-encoded form or as their numeric values, depends on the transmission mode which, in turn, is determined by what the MCS application expects to process.

As a general rule, where the graphic symbol exists for the code, MCS processes both forms in the same manner, translating the code into its numeric value.

A numeric value, not corresponding to a displayable code, is output in normal mode to the terminal in character-encoded form.

The "repeat" function is performed where the character-encoding marks specify 2 valid hexadecimal digits, and in both cases, that is, character-encoded form and numeric value, the code is repeated in its numeric value.

Data Representation Using Mark Form (Character-Encoding & Repeat)

To send text [Date] and a string of 80 asterisks [*]

Refer to the EBCDIC table for the EBCDIC values of the characters.

D = C4	a = 81	t = A3	e = 85	* = 5C

MCS COBOL

a. For "Date", use either form of coding :

```
77 DAT PIC X(16) VALUE "D><C81><CA3><C85".
```

```
77 DAT PIC X(20) VALUE "><CC4><C81><CA3><C85".
```

b. For string of 80 asterisks, use either form of coding :

```
77 AST PIC X(6) VALUE "><R80*".
```

```
77 AST PIC X(10) VALUE "><R80><C5C".
```

GPL

a. For "Date", use either form of coding :

```
DCL DAT CHAR(16) INIT ("D><C81><CA3><C85");
```

```
DCL DAT CHAR(20) INIT ("><CC4><C81><CA3><C85");
```

b. For string of 80 asterisks, use either form of coding :

```
DCL AST CHAR(6) INIT ("><R80*");
```

```
DCL AST CHAR(10) INIT ("><R80><C5C");
```

Control Codes

Control codes are generated by the terminal when the touch-key representing the appropriate control function is pressed.

The programmer, however, encodes these control functions to send to the terminal in order to activate certain terminal management functions.

While in the majority of cases, the control function is associated with a single control code, other more complex control functions are implemented by a control code sequence, represented by a combination of control codes and/or graphic symbols.

If the user is not concerned with control codes generated by the terminal and only wants to activate basic editing functions when sending messages to the terminal, he should specify the normal mode for both input and output transmission.

On input, all control codes will be suppressed from the message text by MCS.

On output, the user may activate basic editing functions by specifying,

- the AFTER "advancing" PAGE clause of the [$H_]SEND verb
- MCS automatic editing functions.

AFTER ADVANCING PAGE

When the AFTER "advancing" PAGE clause is used with the last [$H_]SEND which terminates the message with either EMI or EGI, MCS then automatically generates control codes for insertion before and after the message text according to

- the type of the device receiving the message
- the control function for the type of terminal management requested.

In the programming example facing the page, the following actions are performed on a VIP7700 terminal,

- to build the screen line by line
- to generate a "form-feed" function on the last line of the message.

The following considerations are to be taken into account when coding, namely,

- the VIP7700 automatically performs a "new-line" function at the end of each line
- the AFTER "advancing" PAGE in the last line of the message generates the "form-feed" function, a service provided for by MCS.

The text referred to, to be moved into the output buffer OUTBUF, can be any text either for formatting the screen or for displaying form entries.

Alternative forms of programming are given in both MCS COBOL and GPL examples, both of which cater for filling in the entire standard screen of the VIP7700, being 24 lines of 80 characters.

MCS COBOL

```
DATA DIVISION.
  77 OUTBUF PIC X(80).
  77 IDX COMP-1.
COMMUNICATION SECTION.
  CD CD-OUT OUTPUT
    only user-initialized CD-output parameters are shown
    DESTINATION COUNT  COUNT-OUT
    TEXT LENGTH        LENGTH-OUT
    DESTINATION        DESTINATION-OUT.
PROCEDURE DIVISION.
  MOVE 0 TO IDX.
  MOVE 1 TO COUNT-OUT.
  MOVE 80 TO LENGTH-OUT.
  MOVE destination TO DESTINATION-OUT.
  move text into OUTBUF and use either form of coding following
```

```
LOOP23.
  SEND CD-OUT FROM OUTBUF.
  ADD 1 TO IDX.
  IF IDX < 23 GO TO LOOP23.
  SEND CD-OUT FROM OUTBUF WITH EMI AFTER ADVANCING PAGE.
```

```
LOOP24.
  SEND CD-OUT FROM OUTBUF.
  ADD 1 TO IDX.
  IF IDX < 24 GO TO LOOP24.
  SEND CD-OUT WITH EMI AFTER ADVANCING PAGE.
```

GPL

```
DCL OUTBUF CHAR(80);
DCL IDX FIXED BIN(15);
$H_CD OUTPUT , PREFIX='FIRST_';
USER_DESTINATION_COUNT=1;
USER_TEXT_LENGTH=80;
USER_QUEUE_NAME="destination";
IDX=0;
move text into OUTBUF and use either form of coding following
```

```
DO IDX=1 TO 23;
$H_SEND 'ADDR(FIRST_OUTPUT_CD)', INADDR='ADDR(OUTBUF)';
END;
$H_SEND 'ADDR(FIRST_OUTPUT_CD)', INADDR='ADDR(OUTBUF)'
        ENDCHAR=EMI , AFTER , PAGE;
```

```
DO IDX=1 TO 24;
$H_SEND 'ADDR(FIRST_OUTPUT_CD)', INADDR='ADDR(OUTBUF)'
END;
$H_SEND 'ADDR(FIRST_OUTPUT_CD)', INADDR='NULL()'
        ENDCHAR=EMI , AFTER , PAGE;
```

MCS AUTOMATIC EDITING

Automatic editing functions provided for by MCS are activated through the following parameters of the QUEUE command which apply specifically to the terminal-queue and are declared at network generation, namely,

- BLOCKING : MCS keeps track of the logical line-length specified by LLENGTH in order to generate automatically a new-line or carriage-return/line-feed before the first line of each output message or at the start of a new logical line
- LLENGTH : Specifies the number of characters in the logical terminal line length to be used for automatic editing when BLOCKING is specified
- NBLOCKS : Defines the number of logical lines to be accepted in each message sent to the terminal declared, where the number of characters for each logical line is determined by LLENGTH.

If the message is greater than NBLOCKS x LLENGTH, the characters in excess are truncated.

Representing Control Codes

Control codes are represented by

- numeric values
- mark form.

Control code sequences are a combination of control codes and/or graphic symbols.

NUMERIC VALUES

In the example opposite, the numeric values for the control codes CR∇ and LF∇ are specified in accordance with what is expected by the compiler, namely,

- for the MCS COBOL compiler, the decimal value of the COBOL collating sequence
- for the GPL compiler, the hexadecimal EBCDIC value.

In GPL, no intermixing between graphic representation and hexadecimal values is allowed, and for that reason, control codes to be filled in must first be initialized as spaces in the constant character string.

The example also shows the formatting of the VIP-header and its parameter codes.

The mark form >\<U03 is the invariable VIP-header and can be regarded as a special case of the control code in mark form, which is,

- delivered on input in front of the message text from the terminal
- provided on output in front of the message text by the user
- allowed in the unedited mode, both on input and output, in order for the programmer to access the status and function codes.

Control Code Sequence Using Numeric Values

To send to the VIP7700 printer the text format

```
BOOKINGS
NUMBERS:
```

1. To address the VIP7700 printer, use the following values for the VIP-header.

   STA = 3F (EBCDIC value)
   = 64 (COBOL Collating Sequence)

   FC1 and FC2 to be left initially as spaces

   The format of the VIP header is

>	<	U	0	3	STA	FC1	FC2

2. To position the hard copy for the second line of text, use the values:

   CR∇ = 0D (EBCDIC value)
   = 14 (COBOL Collating Sequence)

   LF∇ = 25 (EBCDIC value)
   = 38 (COBOL Collating Sequence)

3. In the MCS application, code as appropriate :

   MCS COBOL

   For MCS COBOL, use the COBOL collating sequences

   ```
   DATA DIVISION.
   77 HEAD PIC X(26) VALUE ">><U03"64"∇∇BOOKINGS"14"
                           "38"NUMBERS:".
   ```

   GPL

   For GPL, use the EBCDIC values

   ```
   DCL HEAD CHAR(26) INIT (">><U03∇∇∇BOOKINGS∇∇NUMBERS:") ;
   SUBSTR(HEAD,6,1) = "3F"H ;
   SUBSTR(HEAD,17,2) = "0D25"H ;
   ```

MARK FORM

The mark form enables any control code, known to the "stream processor", to be entered in its mnemonic form.

This easy-to-use symbolic representation of control codes is a general facility of MCS.

For the control code sequence, the 2 types of mark form dealt with are,

- ><ccc, where ccc represents the 3-character mnemonic of the control code
- ><C, denoting character-encoding for the EBCDIC numeric value to follow, thereby completing the control code sequence.

The "repeat" function is not treated here, since the processing of "repeated" control codes, in whatever form, is specific to the line procedure, see "Control Code in Mark Form" and "Character-Encoded Form" on page 3-04.

The choice of entering control codes either in their character-encoded form or in their mnemonic mark form, depends on the type of control code.

In general the control code or the control code sequence is expressed in character-encoded form, under the following conditions,

- when the control code is not defined as standard, see "Control Codes"
- when the control code sequence is composed of data which individually is neither control codes nor graphic symbols, and therefore, cannot be specified in any other form.

Control Code
Sequence
in Mark Form

To position the cursor of the VIP7700 screen on line 11 at column 37.

1. Refer to the VIP7700 terminal manual for the format of the command to position the cursor.

   command is DC3ab, where, a is the line number
   b is the column position

2. Refer to the ASCII table to determine the values for line and column.

   To determine the EBCDIC value for line 11, proceed as follows,

   - Start from ASCII code 20 which is a "space"
   - Count 11 codes from the ASCII code 20
   - The ASCII code arrived at is 2A or graphic *
   - The EBCDIC equivalent is 5C

   To determine the EBCDIC value for column 37, proceed as follows,

   - Start from ASCII code 20 which is a "space"
   - Count 37 codes from the ASCII code 20
   - The ASCII code arrived at is 44 or graphic D
   - The EBCDIC equivalent is C4

3. In the MCS application, use either coding:

MCS COBOL	77 CONT PIC X(7) VALUE "><DC3*D". 77 CONT PIC X(15) VALUE "><DC3><C5C><CC4".

GPL	DCL CONT CHAR(7) INIT ("><DC3*D"); DCL CONT CHAR(15) INIT ("><DC3><C52><CC4");

SECTION IV

CONNECTION HANDLING

Connection handling describes the dynamics of establishing the connection between users, represented by terminals and applications.

The connection interface between both local and remote users, is assured by the Message Control System, whose functions are provided by MAM and QMON.

Once the connection has been established, data exchange can then take place.

Whereas the connection in the case of VCAM subsystems is direct, the connection in the case of MCS applications is logical, in order to allow for the following conditions,

- the application can be connected to an output queue whose destination terminal is not active
- the terminal can be connected to an input queue whose associated application is not executing.

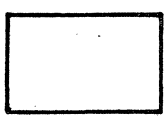
The term "local terminal" refers to a terminal connected over secondary networks, that is, both the local and the TRANSPAC secondary networks accessed over BTNS. The logon for terminals configured over these secondary networks is treated in detail in the Terminal Operations Manual.

The term "remote application" refers to an application executing in a machine other than the DPS 7 local system. This other machine can be connected to the local DPS 7 either over the secondary network through the BSC2780 line procedure or over the primary network. In the case of the primary network, the local DPS 7 acts either as the "host" over the FNPS/DN7100 interface or as the "satellite" over the TNS/URP interface.

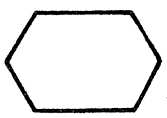
A comprehensive description of Distributed Systems Architecture and DPS 7 networks is given in the Communications Overview Manual.

This section is intended to describe how the various types of network connections are handled and the effects of the $QASSIGN statements in each case. For details on $QASSIGN, see pages 2-05 through 2-53.

The conventional symbols in the text are as follows,



GCOS process
- BTNS and FNPS
- QMON



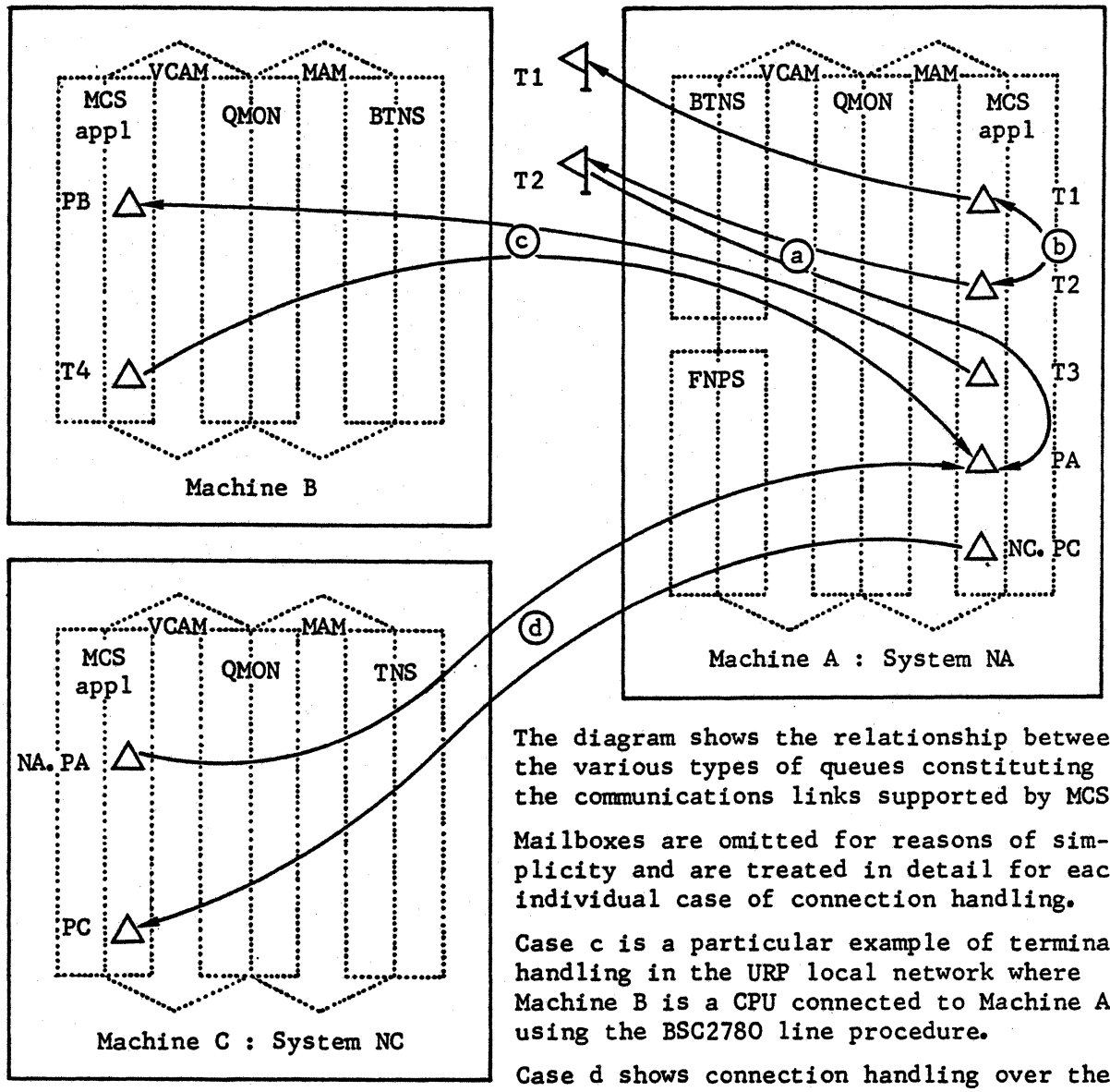
GCOS access method
- MAM
- VCAM

terminal mailbox as "end-point"

application mailbox as "end-point"

MCS queues
- T terminal-queue ; P program-queue
- Nx. Py DSA-queue <system. queue>

Communications Links
supported by the
Message Control System



The diagram shows the relationship between the various types of queues constituting the communications links supported by MCS.

Mailboxes are omitted for reasons of simplicity and are treated in detail for each individual case of connection handling.

Case c is a particular example of terminal handling in the URP local network where Machine B is a CPU connected to Machine A using the BSC2780 line procedure.

Case d shows connection handling over the primary network using the FNPS/DN7100 interface of Machine A and the TNS/URP interface of Machine C.

References in the text following, are

(a) see "Manual Logon from a Local Terminal to a Local Application"

(b) see "Logon from one Local Terminal to another Local Terminal"

(c) see "Communication between Remote Applications", Section II

(d) see "Connection Request from a Local Application to a Remote Application"

The term "machine" refers to the DPS 7 either as a system in a DSA primary network or as an HL64 CPU terminal in a secondary network.

The term "application" is used to group together VCAM subsystems and MCS applications.
During the logon of a general-purpose terminal, the operator keys in, in response to APPL, either the name of the VCAM subsystem or the name of the program-queue denoting the MCS application, see "Logon Procedures", Telecommunications Ref. Card.

Besides local applications communicating in the same machine, the Message Control System supports other communications links, shown in the diagram opposite, which are the following, in relation to the machine in which they occur,

- over the BTNS/URP interface in a secondary network

  (a) Machine A :

  - between the application PA, represented by the program-queue PA
  - and the terminal T2, represented by the terminal-queue T2

  (b) Machine A :

  - between the terminal T1, represented by the terminal-queue T1
  - and the terminal T2, represented by the terminal-queue T2

  (c) Machine A linked up to Machine B, using the BSC line procedure :

  - between the BSC terminal T3, represented by Machine A, and the application PB, situated in Machine B
  - and, between the BSC terminal T4, represented by Machine B, and the application PA, situated in Machine A

- in a primary network using the FNPS/DN7100 interface of Machine A on the one side, and the TNS/URP interface of Machine C on the other side.

  (d) Machine A linked up to Machine C, as systems in a DSA network, where A and C are respectively the systems NA and NC :

  - between the DSA-terminal-queue NC.PC, representing Machine A as system NA, and the application PC situated in Machine C as system NC
  - and, between the DSA-terminal-queue NA.PA, representing Machine C as system NC, and the application PA situated in Machine A as system NA

Connection handling is dealt with in terms of

- connection request from a local terminal
- connection request from a local application
- connection request from a remote application.

For local applications, see "Communication between Local Applications", see Section II.

CONNECTION REQUEST FROM A LOCAL TERMINAL

The cases considered for connection requests from a local terminal are,

- manual logon from a local terminal to a local application
- logon from an automatic-dedicated terminal to a local application
- logon from an automatic terminal to the QMON mailbox
- manual logon to a "blank" destination or to the QMON mailbox
- logon from a local terminal to another local terminal.

Manual Logon from a Local Terminal to a Local Application

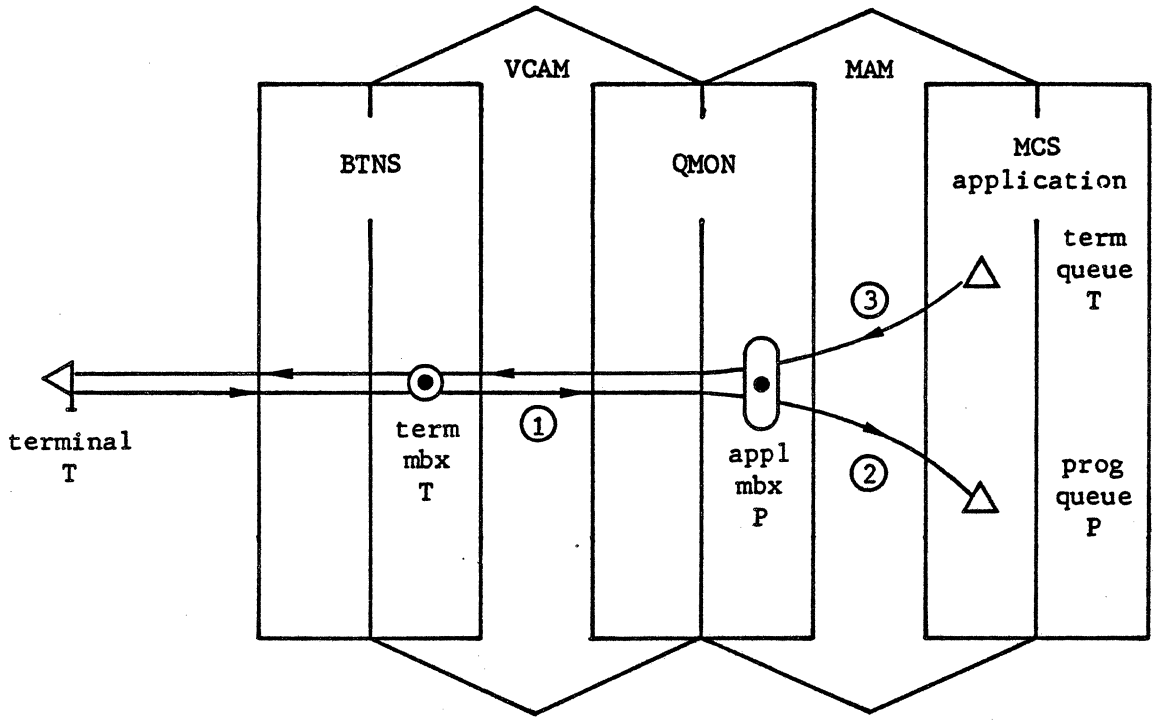
The local terminal referred to in this instance is a general-purpose terminal, that is, declared at network generation with neither AUTO nor ASSIGN.

Such a general-purpose terminal can communicate with any MCS application through a program-queue defined at network generation.

The connection is rejected if the following anomalies occur,

- CCO4 LOGON DENIED : SECURITY CHECKS FAILED
  - one or more of the specified catalog parameters, for a validated GCOS site-catalog, given in reply to the CCO1 message is incorrect.
- CCO4 LOGON DENIED : APPL REJECT
  - the specified MCS program-queue has neither been defined nor enabled, or is saturated at the time of the connection request.
  - one of the following terms has not been defined,
    - a terminal-queue bearing the same name as the terminal-mailbox
    - a userid-queue bearing the same userid of the terminal operator.
  - the terminal-queue or userid-queue is not available since some executing application has a $QASSIGN OUT on the terminal concerned but does not have a $QASSIGN IN on the designated program-queue.

Manual Logon
from a Local Terminal
to a Local Application



JCL for the MCS application, if executing at logon time, must contain :
- $QASSIGN IN on the program-queue P
- and, optionally, $QASSIGN OUT on the terminal-queue T

1.a Terminal T requests connection through manual logon to the MCS program-queue P, specifying P as the application.

1.b The BTNS terminal handler initiates the logical connection between the terminal-mailbox T and the application-mailbox P, which has the same name as the MCS program-queue P.

1.c The connection is accepted by QMON when all conditions are satisfied, that is,
- if catalog access rights have been validated
- if the program-queue P is defined, enabled and not saturated
- if the terminal-queue or userid-queue is defined and available.

1.d Data exchange can now take place.

2. Messages sent by the terminal T are placed by QMON into the program-queue P and from there retrieved by the MCS application for processing.

3. Messages sent by the MCS application are placed in the terminal-queue T and from there are retrieved by QMON for transmission to the terminal T.

While the connection is established, any assign request on the terminal- or userid-queue through $QASSIGN OUT, will be rejected.

Logon from an Automatic Dedicated Terminal to a Local Application

An automatic-dedicated terminal is one where both the AUTO and ASSIGN options have been declared at network generation.

The connection request is automatically handled by the BTNS terminal manager which acts on behalf of the terminal as soon as the terminal is powered on and no HT network control command has been previously issued to the terminal or to any component of the link.

The userid generated by the secondary network controller for the terminal has the syntax <gencom-name><terminal-name> or <lsys-name><terminal-name>.

In order for catalog access rights to be established for such a terminal, the project of this userid must specify in its CRP command of the CATMAINT utility, the program-queue in its APPLIST, see System Management Guide.

The only logon dialog is in the case where the terminal is connected over a switched line and declared with the IDSEQ command at network generation.
The BTNS secondary network controller then sends the message CC00 ID ?, to which the operator replies with the appropriate id, specified in the IDSEQ command.

Logon from an Automatic Terminal to the QMON Mailbox

An automatic terminal is one where only AUTO has been declared, and not ASSIGN.

Such a terminal will be placed in the "logged" state by the BTNS terminal manager and will then be available for allocation to any application which requests it.

Before being placed in the "logged" state, however, the connection request is addressed by BTNS to QMON, to check if there is any output available in the terminal-queue.

If the terminal-queue has data, and is enabled, the connection is then established between the terminal-mailbox and the QMON-service-mailbox, QMONMBX.
Once all data has been released to the terminal, and if no $QASSIGN OUT is pending on the terminal-queue, the connection is broken, and the terminal is set back to the "logged" state.

If, however, the conditions for output to the terminal are not satisfied, the terminal will immediately be placed in the "logged" state.

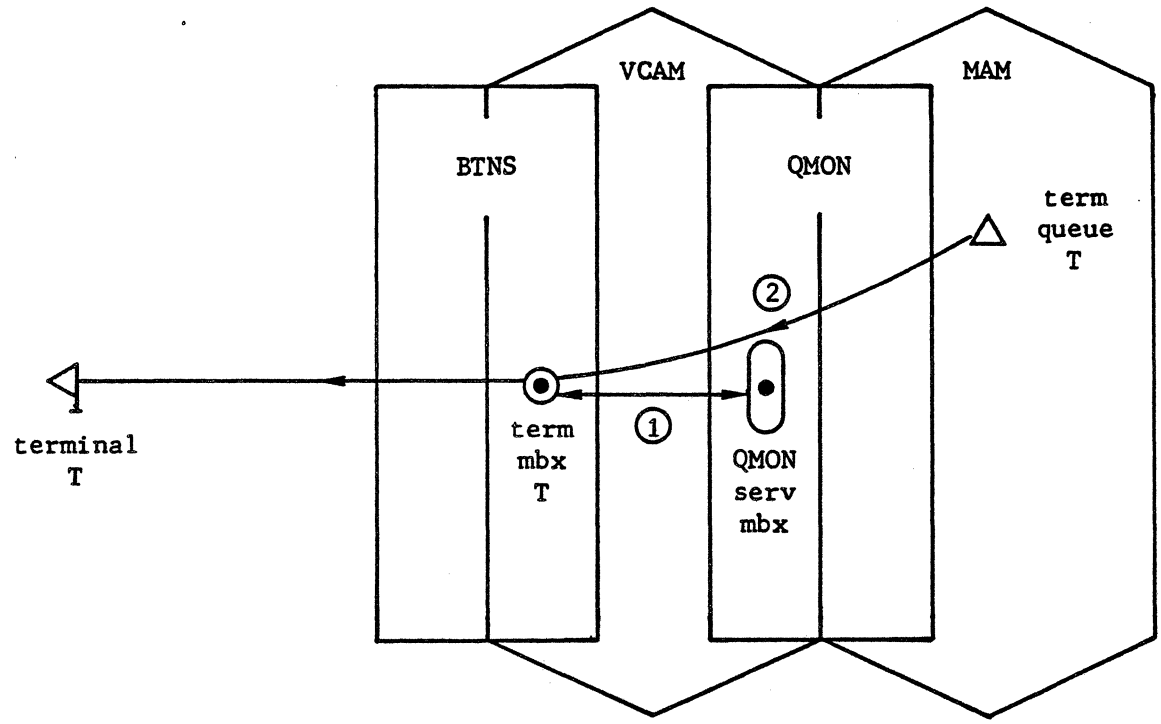
As in the case of the automatic-dedicated terminal, the only logon dialog occurs for a terminal connected over a switched line and declared with the IDSEQ command at network generation.
The BTNS secondary network controller then sends the message CC00 ID ?, to which the operator replies with the appropriate id, specified in the IDSEQ command.

Manual Logon to a "Blank" Destination or to the QMON Mailbox

The difference between this case and the preceding automatic terminal, is

- while the terminal is set to the "logged" state, if no output is available,
- the terminal is set to the "idle" state, after data has been output to it.

Manual Logon
to a "blank" Destination
or to the QMON Mailbox



This type of connection is initiated by one of the following operator actions,

- where no application has been specified
- where QMONMBX, being the system reserved name of the QMON service-mailbox, has been specified as the application.

1. Where no application has been specified at logon, the connection between the terminal-mailbox T and the QMON service-mailbox, QMONMBX, will be established, when both conditions are fulfilled, namely,
   - if a default application has not been specified for the project
   - if there are messages available in the terminal-queue T.

   Where QMONMBX has been specified as the application, the connection will be established when the second condition is fulfilled.

2. Once the connection is established, messages are transmitted from the terminal-queue T to the terminal T.

   The connection is terminated, when all messages have been sent to the terminal, and no $QASSIGN OUT on the terminal-queue T is pending.

Logon from one Local Terminal to another Local Terminal

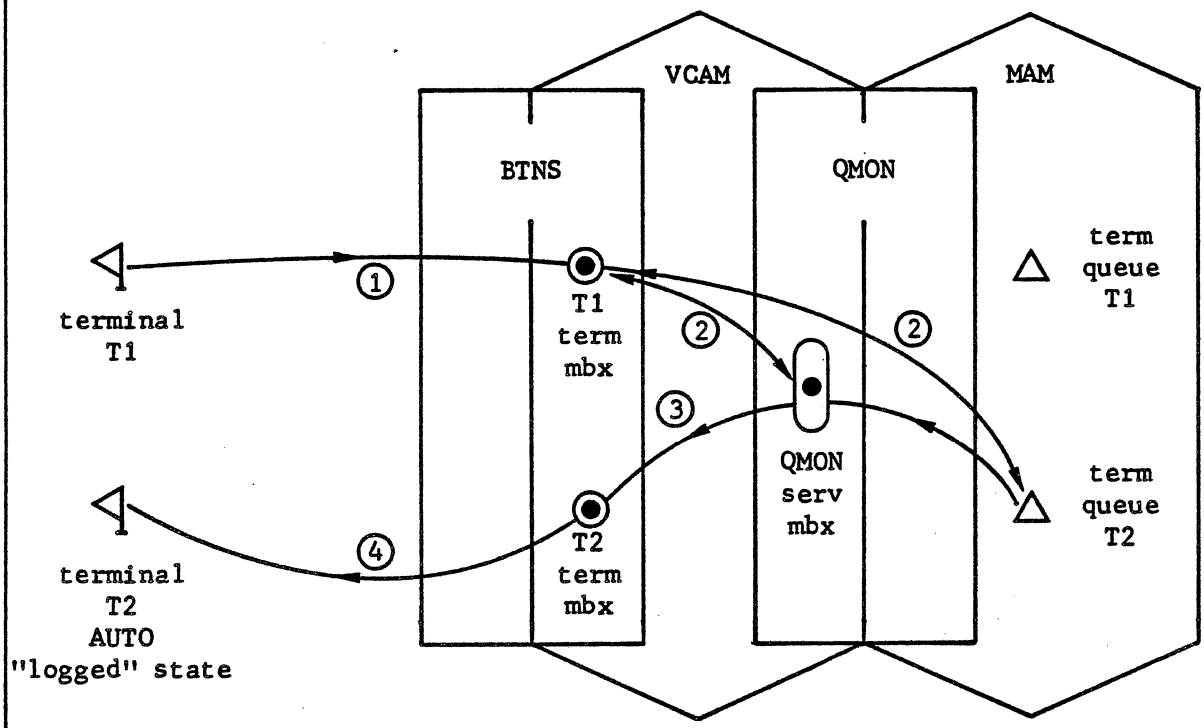
Logging a terminal onto another enables data to be exchanged between the two terminals concerned.

The terminal-queue of the terminal receiving the data becomes the input queue for the terminal sending the data.

In the diagram opposite, the terminals have the following attributes,

- T1 is a general-purpose terminal, which at logon, is able to specify T2 in reply to the option APPL
- T2 is either an automatic or automatic-dedicated terminal, which at logon is set to the "logged" state, in order to receive the data placed in its terminal-queue by T1.

Logon
from one Local Terminal
to another Local Terminal



Conditions : a. Terminal-queues T1 and T2 must both be defined and available.

b. There must be no $QASSIGN OUT pending on either terminal-queue, T1 or T2.

c. Terminal T2 must be in the "logged" state.

1. At logon, terminal T1 specifies T2 as the application, where T2 also serves to identify the terminal-mailbox as well as terminal-queue.

2. The first connection is made between the terminal-mailbox T1 and QMONMBX, being the system reserved name of the QMON service mailbox.

   If terminal T2, at this stage is not "logged", only this first connection is made, thus allowing terminal T1 to send messages to terminal-queue T2.

3. The second connection is then made between QMONMBX and the terminal-mailbox T2, if terminal T2 is "logged", when the first message is sent from T1 to T2.

4. Once these 2 connections have been established, data submitted by terminal T1 into terminal-queue T2, will be retrieved by QMON for transmission to terminal T2.

CONNECTION REQUEST FROM A LOCAL APPLICATION

The connection between two local applications is dealt with in Section II, which outlines the communication between two applications in the same central processor, namely,

- application-to-application communication
- application communicating with itself
- communication between processes of a multiprocess application.

In this section, the two cases considered for connection requests from a local application are,

- to a local terminal
- to a remote application.

Connection Request from a Local Application to a Local Terminal

The network prerequisites are that the terminal T, its terminal-queue T and the program-queue P must be declared by the TERMNL and QUEUE commands, one for each queue, respectively, during network generation (CNC).

The connection is established between the terminal-mailbox T and the application-mailbox P, when all events occur,

- the application P sends the first message to the terminal-queue T
- the terminal-queue is enabled
- the terminal is in the "logged" state, for one of the following reasons,
  - it is either an automatic or automatic-dedicated terminal, which at logon will be set to "logged"
  - it is a general-purpose terminal "logged" on to either a "blank" destination or to the QMON-service-mailbox, QMONMBX.

Data transfer can only occur when the connection is established.

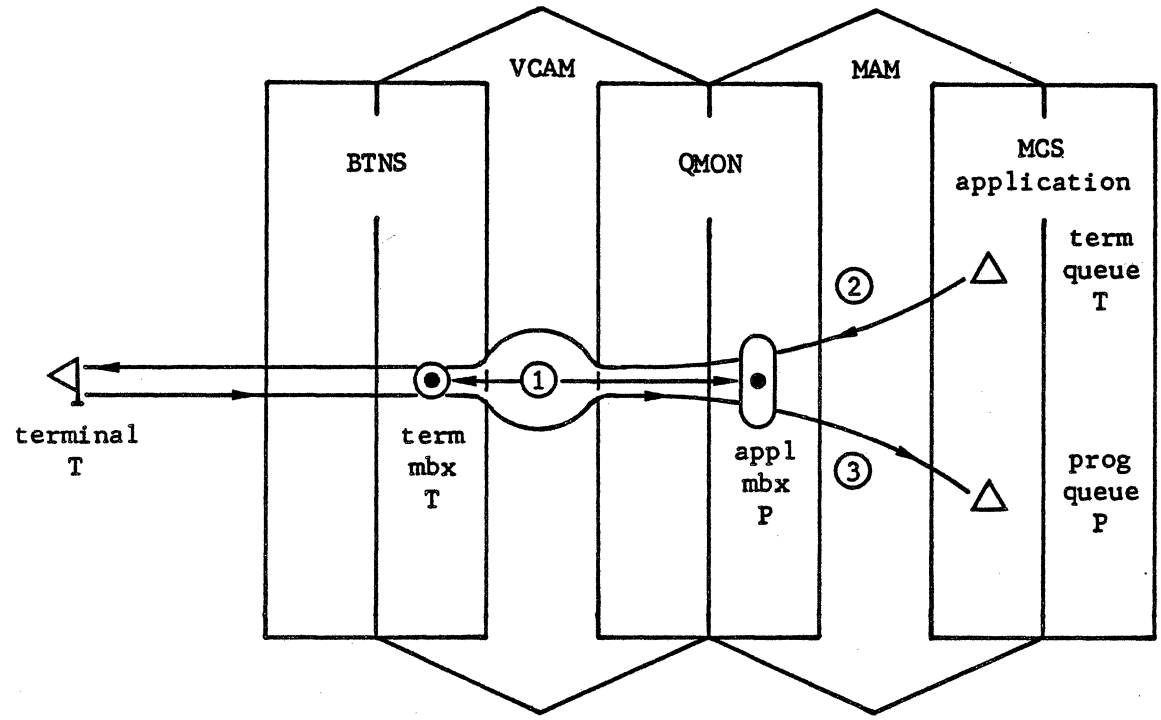
Messages will be placed by the application into the terminal-queue, and will remain in the terminal-queue, if the connection cannot be made for either reason,

- the terminal-queue was disabled, in which case, the connection attempt will be made when a ($H_)ENABLE OUTPUT is executed on the terminal-queue
- the terminal was not in the "logged" state, in which case, the connection attempt will be made when the terminal is set to "logged" from any unapplicable state, such as "powered-off", "held" or "idle".

Disconnection will take place when one of the following conditions occur,

- at the request of the terminal, namely, logoff
- on the failure of any component constituting the physical link
- on termination of the application and when the terminal-queue has released all its messages to the terminal.

Connection Request
from a Local Application
to a Local Terminal



JCL for the MCS application must contain

- $QASSIGN OUT on the terminal-queue T
- and, optionally, $QASSIGN IN on the program-queue P, see 1.b and 3.b.

1.a The connection between the application-mailbox P and the terminal-mailbox T is established for data transfer where both conditions defined above are fulfilled.

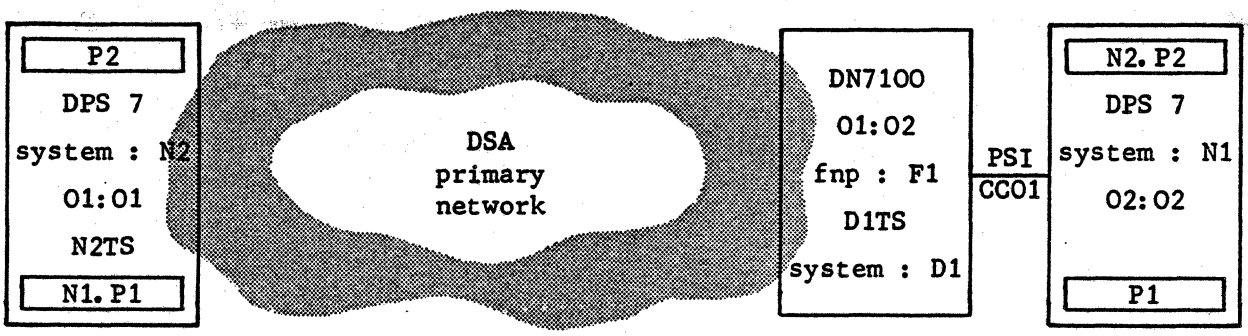
1.b If, however, no $QASSIGN IN has been declared on the program-queue P, the connection will instead be established between the QMON service mail box QMONMBX and the terminal-mailbox T.

2. Messages sent by the MCS application are placed in the terminal-queue T and from there are retrieved by QMON for transmission to the terminal T.

3.a If the connection described for 1.a has been established, messages sent by the terminal T are placed by QMON into the program-queue P.

3.b If the connection described for 1.b has been established, the terminal T functions as a "receive-only" terminal.

Connection Request from a Local Application to a Remote Application (CNC Declaration and Link-up)



CNC Declaration for System N2

```
QUEUE  P2... ;
QUEUE  N1.P1... ;

LSYS  N2 ;
LSC  SCID=01:01  TS=N2TS ;

RSYS  D1 ;
RSC  D1  SCID=01:02  TS=D1TS ;

RSYS  N1 ;
RSC  N1  SCID=02:02  TS=D1TS ;
```

CNC Declaration for System N1

```
QUEUE  N2.P2... ;
QUEUE  P1... ;

LSYS  N1 ;
LSC  SCID=02:02 ;

FNP  F1  CCO1 ;

FSYS  D1 ;
FSC  D1  SCID=01:02  FNP=F1 ;

RSYS  N2 ;
RSC  N2  SCID=01:01  FNP=F1 ;
```

° The conditions for the connection request to be successful are,

. For System N1,
- its front-end processor must be "loaded" and its FNPS service must be activated through the "MTF D1 AUTO" network control command
- its program-queue P1 and DSA-queue N2.P2 are both "enabled"

. For System N2,
- TNS of its BTNS/HDLC service must be activated through the ST command
- its program-queue P2 and DSA-queue N1.P1 are both "enabled".

° Where the connection is unsuccessful, either application can issue a repeated sequence of ($H_)DISABLE OUTPUT - ($H_)ENABLE OUTPUT verbs to the DSA-queue of its correspondent.

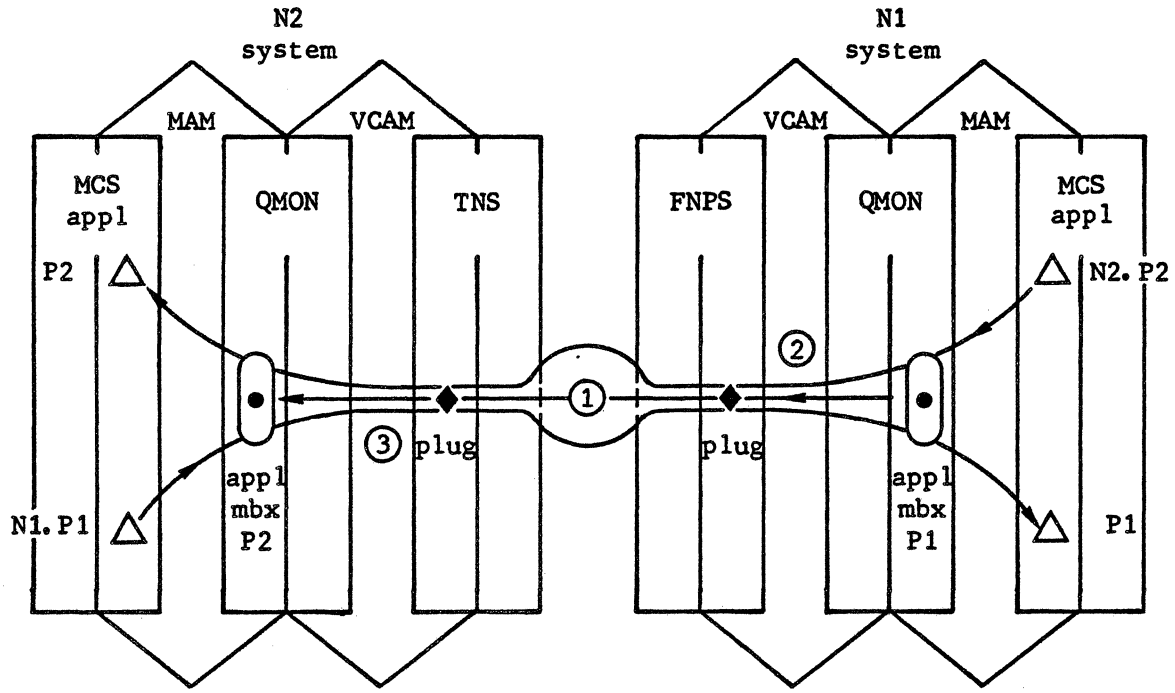
° The connection remains active until terminated by either correspondent,

. on "holding" its program-queue through the HT network control command

. on execution of ($H_)DISABLE INPUT <u>without</u> TERMINAL on its program-queue

. on termination of the application and when its DSA-queue has released all its messages to the other application of its correspondent.

Connection Request
from a Local Application to a Remote Application
(Run-time JCL and Execution)



Conditions :
- For system N1, the JCL for the MCS application P1 must contain
  - $QASSIGN IN on program-queue P1
  - $QASSIGN OUT on DSA-queue N2. P2.
- For system N2, the JCL for the MCS application P2 must contain
  - $QASSIGN IN on program-queue P2
  - $QASSIGN OUT on DSA-queue N1. P1.

1. QMON in both systems N1 and N2 establish the "link-up" between the application-mailbox P1 in system N1 and the application P2 in system N2.

   The connection between the 2 mailboxes in both systems enables data transfers, described in 2 and 3.

2. The MCS application in system N1 sends data from its DSA-queue N2. P2 to system N2.

   QMON in system N2 then places the data received into the program-queue P2.

3. The MCS application P2 in system N2 does likewise in the reverse direction.

   QMON in system N1 then places the data received into the program-queue P1.

CONNECTION REQUEST FROM A REMOTE APPLICATION

Connection handling is the same as that for the connection request from a local application to a remote application.

In this case, however, it is the remote application that initiates the connection request, and as a consequence, the program-queue and the DSA-queue of the local application must be enabled in order for the connection to be successful.

Where the connection is unsuccessful, it is the remote application that must issue a repeated sequence of ($H_)DISABLE OUTPUT - ($H_)ENABLE OUTPUT verbs to its DSA-queue in order to reinitiate the connection request to the local application.

SECTION V

LINE PROCEDURES

Line procedures are used to establish transmission protocols over the link, which comprises the network support for the logical connection of two end-users, namely,

- between the DPS 7 as the local system
- and terminals of the secondary network.

Terminals supported by GCOS are divided into groups, each group corresponding to its specific line procedure.

Terminals within a group are compatible with each other since they conform to a common transmission protocol.

A terminal can, in some cases, be supported by more than one line procedure, for example,

- DTU7171, TN1200 and TTU8124 are supported by the TTY line procedure with the reverse channel version, TTY-R, as an alternative
- BTT7300 is supported by the synchronous and asynchronous versions of the VIP line procedure
- HL64 is supported by both the BSC2780 and the HDLC line procedures.

Terminals are compatible when they function as follows,

- from the point of view of hardware and firmware,
  - they share the same TCT, translation code table
  - they are connected over the same multipoint line when using the "poll/ select" facility
- from the point of view of the application,
  - they share the same controls for formatting the character image.

In this section, the information regarding transmission control codes is as follows,

- where the mnemonic form does not appear in the list of "Control Codes" on page 3-03, the EBCDIC value is given, see pages 5-06 and 5-07
- where the control code is internal to the line procedure, it is shown coded in mark form, acceptable to MCS, see pages 5-13 and 5-17
- where the control code is shown in mark form and is present in the list of "Control Codes", its corresponding EBCDIC value can also be used instead, see "TTY Line Procedure" and Section VI.

The line procedure to which the terminal belongs, determines its operability, namely,

- the controls used to terminate the message
- the special characters used to erase message text.

For this information, see Terminal Operations Manual.

In the following section, "Programming Terminals", the line procedure supporting each terminal is indicated in the heading.

User visibility in programming the terminals listed in Section VI, is limited to the control codes by which the terminals function.

Apart from the BSC line procedure, transmission protocol is seen only at system level.

In the BSC line procedure, the user has access to the transmission control codes in formatting his message.

The following processors, known by their CNC declarations as underlined in the list, operate on the BSC2780 line procedure with the DPS 7,

- HL61 for 61/DPS
- HL62 for DPS 4
- HL64 for DPS 7/xO and DPS 7/x5, where x is a decimal digit
- HL66 for DPS 8/88
- IBM370 for IBM360 and IBM370
- IBM3741.

GCOS communications software provides the user with the facility for data exchange between the DPS 7 local system and any of the central processors listed above, over

- private or leased point-to-point lines, using the BSC1 version of protocol
- switched point-to-point lines, using the BSC2 version of protocol.

The link provides the facility for applications to communicate with each other, see Section IV, "Connection Handling".

As the facility only concerns application-to-application communication, this line procedure does not include automatic processing of end-to-end control codes defined for the BSC2780 IBM-type terminal, namely,

- ESC escape, output device selection
- HT horizontal tabulation, positioning
- EM end-of-medium, variable data length delimiter.

Since these control codes appear within the user text, they can be handled directly by the MCS application.

The procedure is described in terms of

- link states
- logon procedure
- encoding data
- general format of data messages
- management of data transfer by the application
- contents of user messages.

BSC2780
continued

LINK STATES

Except where explicitly defined, the state of the link is completely controlled by the system through BTNS and the URP firmware. The state of the link between two stations using a common communications facility determines their connection and data exchange.

The link can be in one of the following states,

- disconnected state
- control state
- message transfer state.

Disconnected State

The link can be in the disconnected state only in a switched network environment.

This state prevails when

- the physical path is not currently established over the network
- the dial-up procedure is in progress but not yet completed
- the "disconnect" control sequence DLE EOT has been issued when the link has been idle for longer than the inactivity time-out.

The disconnected state is entered from one of the other two states. However, when leaving the disconnected state, the link always enters the control state.

Control State

The link is in the control state when the corresponding physical data path is present but no data transmission is taking place.

The physical data path is established after the successful completion of the connection phase at line initialization.

In the control state, one of the two conditions can occur,

- absence of transmission
- initialization of transmission.

ABSENCE OF TRANSMISSION

When transmission does not take place, the link is idle and the way in which the link is configured in the network determines

- how the network handles the idle condition
- the action taken by the DPS 7.

If the link is supported by a permanent point-to-point type connection, the idle condition persists until the transmission phase is initialized.
When this transmission phase is started, the DPS 7 monitors the line for as long as the link stays open.

If the link is operated over a switched network, the idle condition will be entered from the disconnected state on successful completion of the dial-up procedure.
The DPS 7 will then monitor the line until a "disconnect" control sequence is issued for inactivity time-out.

INITIALIZATION OF TRANSMISSION

The initialization and termination phases of transmission, between which information exchange takes place, are executed in the control state.

Information exchange involves the transmission of control blocks and reply blocks between two stations.

Contention between two point-to-point stations arises when both bid for the line at the same time in order to transmit.

The station wishing to transmit bids for the line by sending the ENQ control code as a transmit request to the other station.
If the request is accepted, the link is brought into the message transfer state and the requesting station can then transmit. Otherwise, the link remains in the control state.
If the request is rejected, bidding for the line is retried up to three times.

BTNS, in its turn, starts bidding for the line each time a new output request is made to it while the line is idle.
If the line is in the logical "open" state, BTNS accepts a transmit request over it.

In order to resolve contention, one of the stations is designated the "primary" station, while the other is designated the "secondary" station.

Attributes of designating priority to the stations are as follows,

- if simultaneous line bids occur, the "primary" station transmit first
- at each bidding, the "primary" station can retry up to three times, and will persist to bid until it receives an appropriate reply : any ENQ control code received by the "primary" station once it has taken action to request the line will be ignored
- the "secondary" station must respond to all valid END control codes that it receives
- before re-issuing the transmit request, the "primary" station has a shorter time-out when waiting for a reply than the "secondary" station, thus forcing both stations out of contention.

According to the URP firmware generation, the HL64 can be defined either as a "primary" or "secondary" station.

INITIALIZATION OF TRANSMISSION (continued)

The functions applicable to the control state for initializing transmission are,

ACKO able to receive, also used for line "turn-around"
EBCDIC 1070

ACK1 able to receive, also used for line "turn-around"
EBCDIC 1061

ENQ can you accept transmission? : also used for line "turn-around"

EOT no synchronization

NAK unable to receive, also used for line "turn-around"

PAD time-fill
EBCDIC FF

RVI you stop transmitting and accept my messages
EBCDIC 107C

SYN start synchronizing, also used as "discard" character

TTD transmission to begin later, respond with NAK or WACK
EBCDIC 022D

WACK request later and wait until acknowledged, also used for line "turn-around"
EBCDIC 106B

Message Transfer State

The message transfer state is a dynamic state which prevails as long as messages and the replies they generate are transferred over the line.

The state is entered at the start of the first message by the start control sequence SOH STX or DLE STX and will be maintained throughout the entire transmission until the last message ended with ETX has been successfully transferred. An end-of-transmission control code EOT is then issued to reset the link to its control state.
The return to the control state can be achieved by the MCS application through KCO or BCO.

In the message transfer state, the station can function in one of two ways,

- as a "master" station, which transmits control codes and block-check's containing redundant information for verification purposes
- as a "slave" station, which receives data and transmits replies.

Station functions are interchangeable and are maintained for as long as the link stays in the message transfer state.

For error free transmission, the "master" station is responsible for resetting the link on termination.
However, if an error condition occurs, whereby transmission can no longer proceed, either station can reset the link to discontinue dialog.

The functions applicable to the message transfer state during transmission are,

ACK0 even block received OK, also used for line "turn-around"
EBCDIC 1070

ACK1 odd block received OK, also used for line "turn-around"
EBCDIC 1061

DLE start-of-transparent-mode

ENQ enquiry, also used for line "turn-around"
- if between blocks, repeat last response
- if at the end of block, ignore block and respond with NAK

EOT cease synchronizing and return to control state, text invalid

ETB end-of-block, also used for line "turn-around"

ETX end-of-text, also used for line "turn-around"

IRS end-of-intermediate-record
EBCDIC 1C

ITB end-of-intermediate-record
EBCDIC 1F

NAK block does not check, retransmit, also used for line "turn-around"

PAD time-fill
EBCDIC FF

RVI message received OK : I would like to transmit
EBCDIC 107C

SOH start-of-block header

STX start-of-text, starts the first block

SYN start synchronizing, also used as "discard" character

TTD transmission to continue later, respond with NAK or WACK
EBCDIC 022D

WACK current block received OK : request later and wait until acknowledged, also used for line "turn-around"
EBCDIC 106B

BSC2780
continued

LOGON PROCEDURE

An easy way to establish the logical connection between a remote station and a program-queue is to dedicate the station to the queue, that is, the corresponding TERMNL command describing the remote station must be declared with AUTO, for automatic logon, and ASSIGN, for dedication to the program-queue, at network generation.

The connection is then established as soon as the line is in the control state and remains until the line returns to the disconnected state.
For an HL64-to-HL64 connection, the link-up may be established on both sides.

If the user wants the remote station to be connected to different applications, that is, to different program-queues, the corresponding TERMNL command describing the remote station must only be declared with AUTO.

In this case, once the remote station is "logged", it will be available for allocation to any application which requests it.

ENCODING DATA

Functionally the BSC2780 line protocol allows two categories of user-provided data to be exchanged over the line, namely,

- text, comprising data to be processed
- heading, containing information for end-to-end control.

Text

Although ASCII encoding is a function of the BSC2780 line procedure provided through the TCTNM parameter of the appropriate LINE command, exchanges between Series 60 processors are performed in EBCDIC code, this internal code being the default option.

For more efficient recovery purposes, text may be split into subblocks separated by sequences of control codes irrespective of the mode of transmission.

Depending on the requirements of the application, text may be transmitted in one of the following modes,

- normal mode
- transparent mode.

NORMAL MODE

Text transmitted must not contain any control codes or control sequences used by the protocol of the BSC2780 line procedure.

TRANSPARENT MODE

The text may contain unrestricted coding of data since all control codes including the "escape" control code DLE must be preceded by DLE in order to be recognized as a control function.

The control functions that can be specified in transparent mode are,

DLE DLE transmits the control code DLE in transparent mode

DLE ENQ forward abort, to denote end-of-transparent-mode

DLE ETB end-of-transmission-block, to denote end-of-transparent-mode

DLE ETX end-of-text, to denote end-of-transparent-mode

DLE ITB end-of-intermediate-block, to denote end-of-transparent-mode

DLE STX start-of-text, to denote start-of-transparent-mode

DLE SYN synchronous idle.

The transparent mode is used when the transmission involves,

- binary data
- floating point numbers
- packed decimal data
- specialized or foreign codes
- computer programs in machine code.

Heading

Heading data is control information used by the application for end-to-end control related to the text data blocks following it.

This control information which is separated from the text in the message, can only be transmitted in normal mode, in one of the following ways,

- embedded within the message containing any type of text, either normal or transparent
- alone in a message, without text to follow it.

BSC2780
continued

GENERAL FORMAT OF DATA MESSAGES

- The format of the data message for transmitting normal text is :

SYN	SYN	SYN	SYN	STX	n-text	ETB	BCC	PAD

SYN	SYN	SYN	SYN	STX	n-text	ETX	BCC	PAD

SYN	SYN	SYN	SYN	STX	n-text	ITB	BCC	SYN	SYN	n-text	ITB	BCC	SYN	SYN	n-text	ETB
																BCC	PAD

SYN	SYN	SYN	SYN	STX	n-text	ITB	BCC	SYN	SYN	n-text	ITB	BCC	SYN	SYN	n-text	ETX
																BCC	PAD

- The format of the data message for transmitting transparent text is :

SYN	SYN	SYN	SYN	DLE STX	t-text	DLE ETB	BCC	PAD

SYN	SYN	SYN	SYN	DLE STX	t-text	DLE ITB	BCC	SYN	SYN	DLE STX	t-text	DLE ETB	BCC
													PAD

SYN	SYN	SYN	SYN	DLE STX	t-text	DLE ETX	BCC	PAD

SYN	SYN	SYN	SYN	DLE STX	t-text	DLE ITB	BCC	SYN	SYN	DLE STX	t-text	DLE ETX	BCC
													PAD

- The format of the data message for transmitting the heading is :

SYN	SYN	SYN	SYN	SOH	heading	ETB	BCC	PAD

SYN	SYN	SYN	SYN	SOH	heading	STX	normal text	ETX	BCC	PAD

SYN	SYN	SYN	SYN	SOH	heading	DLE STX	transparent text	DLE ETX	BCC	PAD

Explanation of Control Codes

SYN SYN SYN SYN
- to establish and maintain synchronization.
  The number of SYN control codes to be inserted by the URP is defined through the ININS parameter of the LINE command at network generation.
  The default number of SYN control codes is 4.

STX
- to denote that the character following is part of the transmitted text.

n-text
- text transmitted in normal mode

t-text
- text transmitted in transparent mode

ETB
- to denote the end of the block and to retain the direction of transmission.
  If the current block is correctly received, another block ended by ETB or ETX must follow.

BCC
- block check control used by URP firmware for checking or generating accumulated parity.

PAD
- used for temporizing between transmissions.
  The number of PAD control codes is defined through the PADNB parameter of the LINE command at network generation.

ETX
- to denote the end of transmission and to allow the reversal in the direction of transmission if EOT ending the last message was correctly received.

ITB
- to denote the end of the subblock and that more subblocks are to follow.

DLE
- data-link-escape used to provide
  - supplementary line control codes in combination with other graphic symbols
    Example : DLE @ is interpreted as RVI (reverse-interrupt)
  - recognition of control functions specified in the transparent mode, see "Transparent Mode".

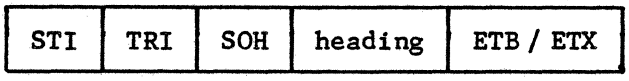
SOH
- to define the start of control information between
  - SOH and STX or ETB in normal mode
  - SOH and DLE STX in transparent mode.

BSC2780
continued

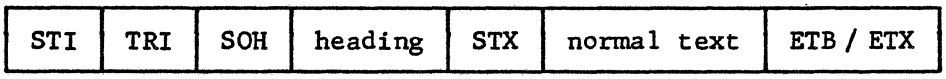
Message Structure seen by BTNS

A text message exchanged between the BTNS message processing routines and the URP firmware can take one of three formats, namely,

- User-defined header message :

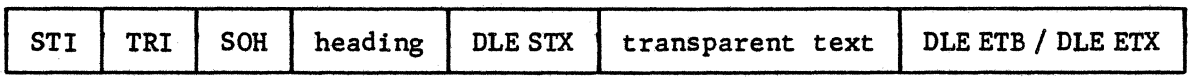


- Message partitioned into blocks :



← optional →

- Transparent message text in which non-standard control codes are accepted :



← optional →

STI
- station index defined by the STATN command at network generation, and set to 1, see Network Generation Manual.

TRI
- terminal index, provision for multipoint handling

SOH
- to define the start of control information

heading
- control information defined and provided by the user

STX
- to denote the start of transmission in normal mode, if not prefixed by DLE.

ETB
- to denote the end of the block transmitted in normal mode, if not prefixed by DLE, and to retain the direction of transmission.

ETX
- to denote the end of transmission in normal mode, if not prefixed by DLE, and to allow the reversal in the direction of transmission

DLE
- to denote transmission in transparent mode, see accompanying codes.

Message Structure seen by the Application

The control codes shown in mark form are translated by the MCS stream processor into their corresponding hexadecimal values and transmitted over the line.

- Format of header on emission and on reception in either mode :

>	<	U	0	4	heading	>	<	U	0	5

- Format of emission in normal mode :

>	<	K	C	0	normal text	ETX	>	<	B	C	0	normal text	ETX

- Format of emission in transparent mode :

>	<	U	0	6	>	<	K	C	0	transparent text	ETX

>	<	B	C	0	transparent text	ETX

- Format of message structure on reception :

>	<	U	0	4	heading	>	<	U	0	5	text

optional

U04
- to denote start of heading in the form of user-defined control information.

U05
- to denote end of heading, and the start of text transmission.

U06
- to denote that the text following is transmitted in transparent mode.

BC0
- communication is "broken", allowing reversal in the direction of transmission after sending the next end-of-file block terminated by ETX

KC0
- communication is "kept", that is, retained, after sending the next end-of-file block terminated by ETX.

text
- user-defined message shown as block transmission terminated by ETX.

BSC2780
continued

MANAGEMENT OF DATA TRANSFER BY THE APPLICATION

Once the link has been established and set in the correct state to transmit data, the station whose turn it is to send data, retains its turn under the following conditions,

- as long as it sends message blocks terminated by [DLE]ETB
- until it sends a block terminated by [DLE]ETX, in which case, the link state can be
  - either retained in the message transfer state so that a new line bid is not necessary for resuming transmission in the same direction
  - or returned to the control state by sending EOT after the last block
- unless RVI has been issued by the receiving station to request the reversal in the direction of transmission.

Unlike other line procedures for which BTNS manages the control codes, the user application, in the case of the BSC2780 line procedure, can control such functions affecting line usage through the MCS interface.

Emission

The user application can control the turn through the EMI and EGI delimiters of the [$H_]SEND verb, as follows,

- any message sent with EMI is dispatched by BTNS over the line with [DLE]ETB, thereby retaining the turn
- any message sent with EGI is dispatched by BTNS over the line with [DLE]ETX, thereby signifying the end of the current transmission.

Although the size of the physical block is determined by the buffering capacity of the receiving station, the user application is not concerned with this constraint since BTNS automatically splits messages longer than the buffering capacity by doing the following,

- sending intermediate blocks of the message text with [DLE]ETB
- sending the last block with either [DLE]ETB or [DLE]ETX according to whether EMI or EGI was specified in the application.

The physical block size is 512 characters, except in the case of the Level 61, which is restricted to 450 characters.

Reception

Enqueuing of blocks depends on the control code terminating it, namely,

- a block received with [DLE]ETB is enqueued by BTNS with EMI, thereby setting the ENDKEY field of the input CD of the block to 2
- a block received with [DLE]ETX is enqueued by BTNS with EGI, thereby setting the ENDKEY field of the input CD of the block to 3.

For message processing on reception, see "Message Delimiters", pages 2-03 and 2-04.

Reverse Interrupt

RVI provides the means by which the receiving station indicates to the sending station that

- the last block sent with [DLE]ETB has been correctly received
- transmission must terminate as soon as possible because
  - the receiving station cannot accept any more data
  - the receiving station requires the turn to transmit a message of a higher priority than the message it is currently receiving.

RECEIVING RVI

The actions taken by BTNS when it receives an RVI in the progress of current transmission are,

- the next block is transmitted with [DLE]ETX, whatever the delimiter of the corresponding message is, that is whether EMI or EGI
- the corresponding terminal-queue from which the block was sent is "disabled"
- the control message ><CTLRVI is sent to the program-queue to which the station is connected if the queue was declared with the BREAK option
- the line is set to "input" to receive data from the requesting station.

To detect RVI, the application needs to check the following status keys,

- value "10", for a "disabled" terminal-queue
- value "9B", for a null-length message as contents of the program-queue.

When such an event is detected, the application must

- either restart transmission by issuing [$H_]ENABLE OUTPUT to the terminal-queue if the status key value is "10"
- and/or purge the terminal-queue with the message ><CTLPRG with EGI.

SENDING RVI

If the DPS 7 application receiving messages wants to stop the sending station, it sends the control message ><CTLRVI into the corresponding terminal-queue.

After the reception of every block ending with DLE ETB, BTNS performs as follows

- checks the corresponding terminal-queue if RVI has been requested by the receiving application
- if RVI has been requested, informs the sending station immediately.

Retaining the Communication

The normal sequence of events in transmission is as follows,

- as a default option, BTNS sends EOT after the last block of a transfer specified with [DLE]ETX
- the link then returns to the control state
- the line, as a consequence must be bidden for, in order for a new transfer to take place.

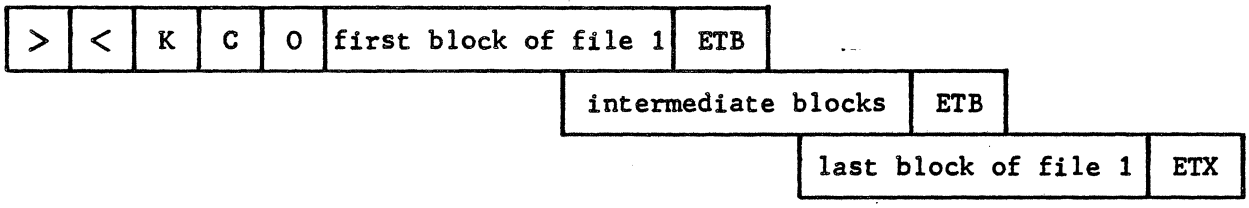
In order to keep the link in the message transfer state, the default option by which BTNS sends EOT, can be overridden.

If several transfers are to take place in the same direction, the application can specify that EOT must not be systematically sent by BTNS by using the following control functions,

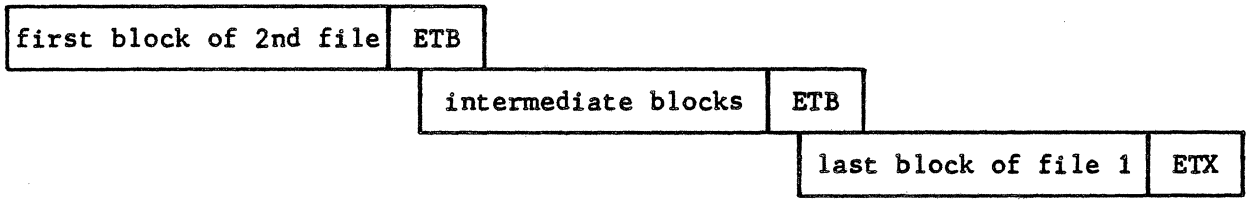
- KCO is specified with the first block of a transfer in order to inform BTNS not to send EOT after the last blocks of the following transfers until a new transfer is initiated with a "break"
- BCO is then specified with the first block of the last transfer in order to inform BTNS that after the last block of the current transfer, it is to send EOT in order to "break" connection and thereby allow the reversal of direction in transmission.

The sequence of transmission at the level of the file, where several files are concerned, is maintained for as long as the turn is kept.

- The 1st file is transferred with KCO specified in the first block :

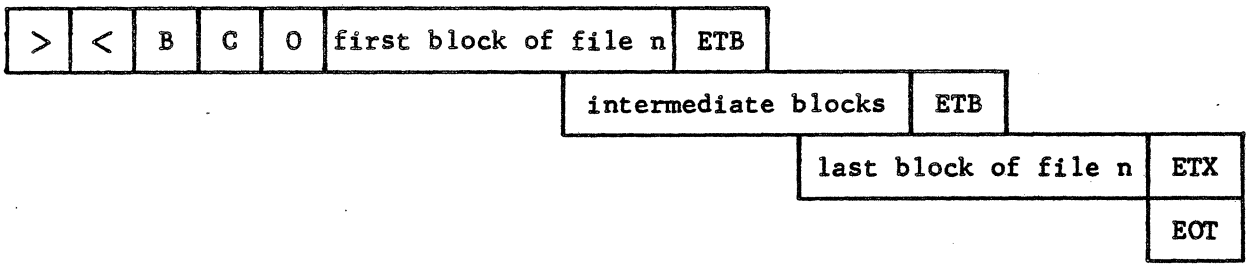


- The 2nd file is transferred immediately after the ETX transmitted at the end of the 1st file :



- All subsequent files are transmitted identically to the 2nd file.

- The nth file is the last file to be transferred and therefore is specified with BCO in the first block :



- EOT is sent only after the last file has been transferred, and until then, the link is still in the message transfer state.

CONTENTS OF USER MESSAGES

The application may send and/or receive heading data alone or heading data followed by the message text.

In either case, the heading character string, limited to 15 characters, is located at the start of the message in the following format.

><U04	heading	><U05	optional normal / transparent message text

The treatment of the message text depends on the mode of transmission.

Treatment of Text in Normal Mode

There is no special format or processing for the text except for the processing of standard marks, such as control codes in mark form, according to

- the IM and OM options declared in the QUEUE command at network generation
- the options issued in the [$*$]MTE network control or terminal operator command during the communications session to override the IM and OM options, respectively.

See Section III, "MCS Data Formats".

Treatment of Text in Transparent Mode

The application receives transparent data in the same way as it receives normal data, except that no processing of standard marks is done, whatever the IM option specified for the corresponding queue.

To send in transparent mode, the user must ensure that the text of each message is preceded by ><U06, in which case, the following actions are taken,

- MCS does not process standard marks
- BTNS provides the appropriate control sequence by prefixing control codes specified in the text by DLE.

The BCS2780 line procedure supports the transmission of files in mixed normal and transparent modes.

Subblocks

Messages are split into subblocks in order to provide a more efficient means of error checking.

Retransmission of the subblock is at the level of the line protocol and as such, is not visible to the application.

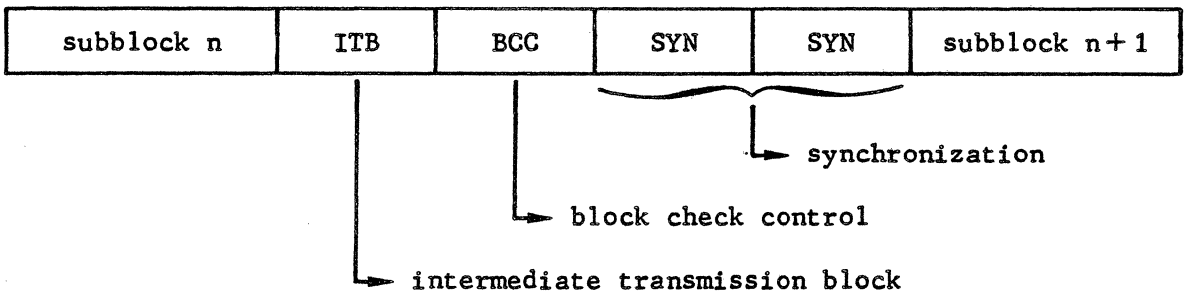
BTNS and the URP firmware ensure that the intermediate control codes separating the subblocks are handled as follows, for error detection and recovery,

- checked for retries
- inserted for emission
- deleted on reception.

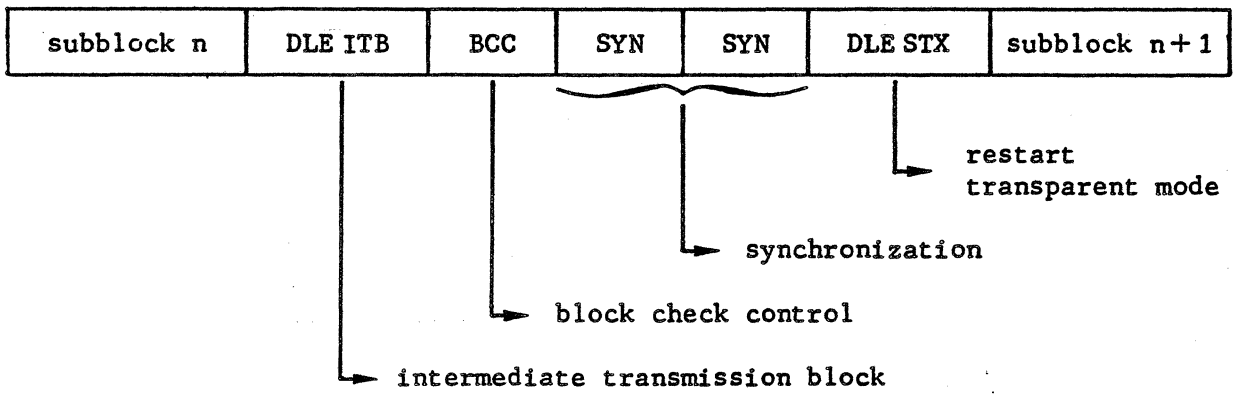
The ability to use the subblock facility over a line is configured at URP firmware generation and is available in both normal and transparent modes of transmission.

The format of the subblock separators depends on the mode of transmission.

- subblock separators in normal mode :



- subblock separators in transparent mode :

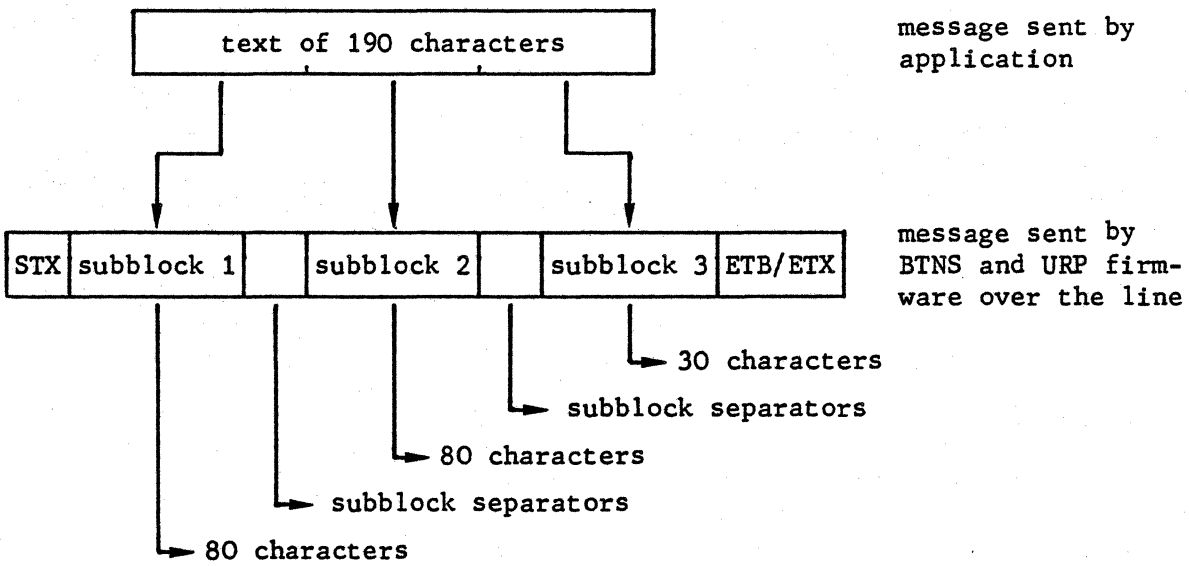


Handling of Text on Emission

The following considerations must be taken into account,

- message blocks are split into subblocks according to the SBKLG parameter of the appropriate TERMNL command, the default value being 80
- the maximum number of subblocks for each message is 16
- a line configured without the subblock option at URP firmware generation cannot use this facility
- even if the subblock option has been specified at URP firmware generation, the facility would not be of much use if the value defined for the SBKLG parameter is greater than any single block sent over the line because, in this case, only 1 subblock per block is defined.

• Blocking of message text on emission :



Handling of Text on Reception

All subblock separators are deleted and the subblocks are handled as contiguous data in the user buffer on [$H_]RECEIVE.

The handling of text on reception is the reverse of that on emission.

The Olivetti TCV260 operates on the TC line procedure when linked to the DPS 7.

Only general information concerning this terminal is treated.

★

MESSAGE FORMAT AS SEEN BY BTNS

A message text exchanged between the BTNS message processing routines and the URP firmware has the following format :

STI	STX	RFE	ADR	message-text	ETB/ETX

STI station index byte, that is, the station address as specified in the STATN command at CNC generation

STX start-of-text

RFE reserved for future extension, based on terminal cabling

ADR address of auxiliary output device, that is, the "terminal-subtype", attached to the terminal

ETB end-of-transmission-block

EBCDIC 26, COBOL Collating Sequence 39

ETX end-of-text

EBCDIC 03, COBOL Collating Sequence 4

MESSAGE TERMINATING CHARACTERS

The control codes, shown in mark form, ><ETB and ><ETX are used interchangeably to indicate the termination of the message text.

The characters are affixed automatically when the "transmit" key is pressed.

Some devices are equipped with ETB and ETX control keys, and when pressed, generate the appropriate control codes.

TC
continued

ERASING FUNCTIONS

Since the terminals are buffered, erasing functions are performed locally through available editing keys, such as "backspace", before the message is transmitted.

TERMINAL SPECIFICS

Refer to respective product manuals.

TTY

The following terminals, known by their CNC declarations, operate on the TTY line procedure when linked to the DPS 7,

- AJ832
- DTU7171
- OLIV318
- TN300
- TN1200
- TTU8124
- TTU8126
- TTY33
- TTY35
- TTY37
- TTY38
- VIP7100
- VIP7200
- VIP7802

These terminals using the TTY line procedure have limited functions in that they have no specific control codes.

Communication is performed through a character-by-character mode over a point-to-point line with the DPS 7.

Messages seen by BTNS contain,

- message text
- one trailer control code on input, in some cases.

MESSAGE HANDLING ON INPUT

On input, messages are terminated when any of the following control codes, shown in mark form, are received, namely,

><CR▽ carriage-return, automatically suppressed by the URP firmware and not received by BTNS

><DC3 paper tape reader stop

><EOT end-of-transmission, with the following considerations,

- automatically suppressed by the URP firmware and not received by BTNS
- available only on terminals equipped with "reverse channel" option and connected over a 1200 bps line, namely,
  - DTU7171
  - TN1200
  - TTU8126

><LF▽ line feed

><SUB substitute, automatically suppressed by the URP firmware and not received by BTNS

The following editing functions are,

- graphic \ = ASCII 5C, used to erase the previous character of the message text, if the LINE connecting the terminal has been declared with the ERCAP option
- graphic @ = ASCII 40, used before "carriage-return" in order to delete the complete line, and hence no message is delivered to the application

TTY
continued

MESSAGE HANDLING ON OUTPUT

On output, messages are terminated on exhaustion of the message length or are edited into fixed length lines, according to the following parameters declared in the QUEUE command,

- LLENGTH
- NBLOCKS
- BLOCKING.

These parameter values can be altered during the communications session either by the network control operator or by the terminal operator, see Terminal Operations Manual.

Other messages are edited by the application before they are submitted to BTNS.

ERROR HANDLING

A limited error recovery capability is possible with terminals using the TTY line procedure.

When the message "RECV ERROR PLEASE REENTER" appears on the terminal, the operator keys in the message text again.

Certain terminals replace invalid characters received from the host computer by a dedicated graphic symbol, e.g., TTU8124 uses the graphic symbol ◇.

Further recovery procedures must be supported by the application.

The following terminals, known by their CNC declarations, operate on the VIP line procedure when linked to the DPS 7,

- BTT7300
- DKU7007
- HL6
- KDS7255 / 7275
- MTS7500
- MTS7508
- STS2840
- TTS7800
- TTU8221
- VIP765
- VIP775
- VIP785
- VIP7001
- VIP7700
- VIP7760
- VIP7804

MESSAGE FORMAT AS SEEN BY BTNS

A message text exchanged between the BTNS message processing routines and the URP firmware has the following format :

STI	SOH	ADR	STA	FC1	FC2	STX	message-text	ETB/ETX	EOT

STI station index byte, that is, the station address as specified in the STATN command at CNC generation

SOH start-of-header

EBCDIC 01, COBOL Collating Sequence 2

ADR terminal address as specified in the TERMNL command at CNC generation

STA status-code, used as follows,

- STA = EBCDIC 00 = NUL, for text only
- STA = EBCDIC 3F = PRT, for print message text

FC1 function-code-1

FC2 function-code-2

STX start-of-text

EBCDIC 02, COBOL Collating Sequence 3

ETB end-of-transmission-block, only used for loading BTT7300

EBCDIC 26, COBOL Collating Sequence 39

ETX end-of-text

EBCDIC 03, COBOL Collating Sequence 4

EOT end-of-transmission

EBCDIC 37, COBOL Collating Sequence 56

VIP
continued

VIP HEADERS

The header of each message sent to or from a VIP terminal contains 3 bytes of information which may be useful to the MCS application.
These 3 bytes are STA, FC1 and FC2. BTNS has no effect on FC1 and FC2, and as a consequence, both these function codes are unedited either on input or on output. The URP firmware uses the TCT, terminal control table, for transcoding them.

The VIP header is visible to the application when one of the following occurs,

- the TERMNL command is specified with either IM=MK or IM=UN
- the network control command MTE specifying the terminal and either IMARK or INEDT is issued
- the terminal operator command $*$MTE specifying either IMARK or INEDT is issued.

The control codes received by the application when one of the above conditions is met, are of the format ><U03abc, where

- a represents STA, the status code
- b represents FC1, function code 1
- c represents FC2, function code 2.

The user can also include message headers of the same format in output messages in normal or unedited mode.

See page 3-37, for programming example of the VIP header.

VIP MESSAGE TRAILERS

Terminals of the VIP line procedure use the sequence ><ETX><EOT to denote message termination.

★

Some terminals are equipped with ETX and EOT keys which, when pressed, generate the appropriate control code.

VIP ERASING FUNCTIONS

VIP terminals are buffered. All editing and erasing functions can be performed locally before entering a transmission request.

Trailing blanks are suppressed from transmitted messages. Most VIP terminals do not transmit empty messages.

When coding applications, the user should disregard empty messages.

SECTION VI

PROGRAMMING TERMINALS

The information concerns the programming interface for terminals functioning with MCS applications.

For details of transmission modes and program coding in MCS COBOL and GPL, see Section III, "MCS Data Formats".

Control codes are shown in mark form, acceptable to MCS, for ease of recognition when listed in alphabetical order according to their mnemonic form. Where the control code is not present in the list of control codes on page 3-03, its EBCDIC value and corresponding graphic symbol, where applicable, are given, for example, the control codes CSI, DAQ and SGR for DKU7007 on page 6-06.

Advice on program coding in this section, deal only with MCS COBOL, for example, SEND AFTER ADVANCING for MTS7500/7508 on page 6-26 and the mention of COBOL for LINE and PAGE for VIP7001 on page 6-45. In both these cases, the GPL equivalent is to be found on page 3-34 "AFTER ADVANCING PAGE".

Terminals are listed in order of their CNC names, that is, the name declared at network generation.

★

Each terminal is headed by the line procedure on which it operates. Transmission protocol implemented at system level is dealt with in Section V.

For details of terminal performance specifications and special functional characteristics, such as the controller for DKU7007, consult the appropriate product manual.

Programming Terminals in TDS
The rules for programming terminals in TDS are as follows, ° Symbolic representation cannot be used in TDS, except for the VIP header ><U03, which can be output to the VIP-type terminal, see page 3-02. ° If the VIP "status" and "function codes" are to be passed to the transaction program, the corresponding TERMNL command for the VIP-type terminal must be declared with IM=UN, see Network Generation Manual. ° Only the numeric form of the control code is acceptable, e.g., page 3-37, . for the COBOL TPR, the COBOL Collating Sequence as a decimal value . for the RPG transaction, the EBCDIC hexadecimal value within apostrophes in the O-spec to the WORKSTN file, with the file-type in the F-spec (col 15) specifying C for the "master" terminal.

Only common terminals are treated in this section. Terminals of the same line procedure, having common characteristics can emulate each other. An example of this emulation is the DKU7001 which functions equally as efficiently declared under DKU7001 as under VIP7200, as was the case in the previous release.

AJ832 operates on TTY line procedure

The AJ832 is a printer with the optional attachments,

- paper tape unit
- cassette handler.

The keyboard can send any of the 128 ISO codes.

The character set for the printer comprises 96 graphic symbols. The number in this character set excludes the "space" but includes the control codes FS▽ and GS▽ in the ISO graphic set, according to the type of printing wheel mounted on the printer.

The product manual gives the graphic correspondence and equivalence between the various types of printing wheels.

Characters received when a parity error occurs, are signalled as follows when the PAR CHK switch is on,

- an acoustic signal is sounded
- the appropriate character is printed with an *.

Line length depends on the model type and the PITCH switch. The maximum line length, however, is 158 characters.

CONTROL CODES

The following terminal control codes in mark form apply to normal mode and assume ASCII graphics as well as the $*$NAPL / $*$napl communications option,

><BEL acoustic signal to the operator

><BS▽ same line, one position to the left

><CR▽ reset print position to programmed left margin

><ESC0 (zero), reset to initial state

><ESC1 set horizontal tabulation stop at current print position

><ESC2 clear horizontal tabulation stop at current print position

><ESC5 set vertical tabulation stop at current print line

><ESC6 clear vertical tabulation stop at current print line

><ESC7 backspace paper one line down, i.e., reverse line throw

><ESC8 backspace paper half a line down, i.e., reverse half-line throw

><ESC9 advance paper half a line up

><ESC><BS▽ clear all horizontal tabulation stops

><ESCD clear all vertical tabulation stops

AJ832
continued

CONTROL CODES (continued)

><ESCJ see ><SI∇
><ESCK see ><SO∇
><ESCL set left margin at current print position
><ESCM clear margins at positions 1 and 132
><ESCN set to normal mode
><ESCR set right margin at current print position
><ESC= set right margin at current physical limit
><FF∇ throw paper to top-of-page
><FS∇ }
><GS∇ } graphics according to the type of printing wheel
><HT∇ move print to next tabulation stop or end-of-line
><LF∇ advance paper one line up, switch selectable at 3 or 6 lines per inch
><SI∇ stop printing
><SO∇ resume printing
><VT∇ advance paper to next vertical tabulation stop or top-of-page

Other Control Codes

A number of control codes are available for,

- page format control
- plotting of diagrams or drawing curves.

Such codes are to be found in the product manual.

★

DKU7007 ★

DKU7007 operates on VIP line procedure

DKU7007 DISPLAY/KEYBOARD

The screen is organized in 25 lines of 80 characters, the last line being reserved for operator information. Including the "space", 95 different graphic symbols, basically ISO/ASCII set with national options, can be displayed. An entry marker is displayed as an aid in formatting the screen.

The FORMGEN utility is available for formatting the screen. The FORMRTP utility, the Forms run-time package, is then executed.

Text for Display

The control sequences for tabulation, entry marker control and message boundaries are the same as for the VIP7700, with functions additional to the VIP7760-2A, see "Define Area Qualification (DAQ)" and "Select Graphic Rendition (SGR)". Blinking and blanking are supported as options.

FORMS MODE

The following terminal control codes in mark form set the terminal either in forms mode or in normal mode. Forms are defined when in normal mode.

><ESCM enters the terminal into forms mode

><ESCN resets the terminal into normal mode

><FS∇ defines the start of a fixed field, must not be specified with DAQ commands

><GS∇ defines the start of a variable field, followed by a value, as follows,

- 0 the field is right-justified, non-repeated and alphanumeric, to be transmitted and printed
- 1 inhibits printing
- 2 inhibits transmission
- 4 numeric-only field
- 8 indicates a number of a line field to be repeated
- 16 the field is left-justified

Must not be specified with DAQ commands.

><RS∇ terminates a repetitive line field, followed by a repetition code

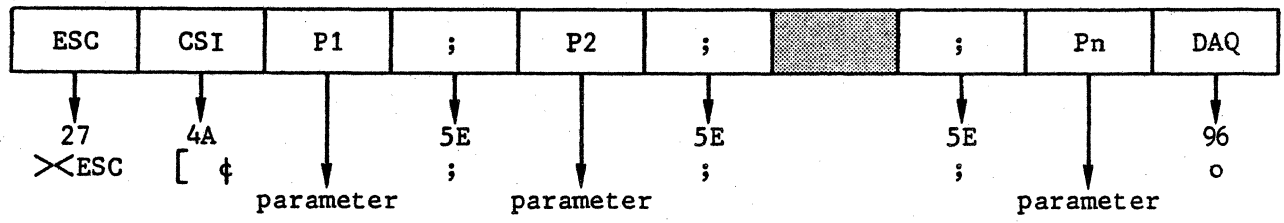
value of repetition code : add 64 to the number of times that the line field is to be repeated.

DKU7007
continued

Define Area Qualification

DAQ commands can only be issued in normal mode. When issued in forms mode, the commands are ignored and no operation results.

Format of command : (values are in EBCDIC, with corresponding graphic symbols)



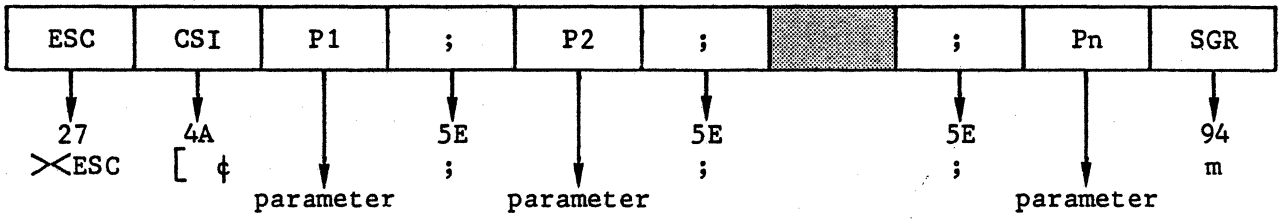
The following authorized parameters can be declared in any order and are in the sequence of EBCDIC values, shown with their corresponding graphic symbols,

4C	<	field entirely filled in by the operator
4CF3	<3	printable field
6E	>	pure decimal numeric field, 0 through 9
7E	=	field that must be filled in by the operator
7EF1	=1	field filled in by the badge reader
F0	0	variable and transmittable field, default parameter if none is specified in the DAQ command
F1	1	fixed and non-transmittable field, being the only authorized parameter for fixed fields
F3	3	decimal numeric field and computational operators, 0 through 9, +, -, ., $
F5	5	right-justified field.

Select Graphic Rendition

SGR commands can be issued in forms as well as normal mode.

Format of command : (values are in EBCDIC, with corresponding graphic symbols)



The SGR command occupies a one "space" position on the screen, hence the entry marker moves forward by one position on completion of the command.

The number of SGR, blink and blank commands is limited to 15 per line.

The following authorized parameters can be declared in any order in the SGR command and are in the sequence of EBCDIC values, shown with their graphic symbols,

F0	0	normal intensity, default parameter if none is specified in the SGR command
F2	2	half intensity
F4	4	underlined by a continuous line
F5	5	blinking
F7	7	reverse video
F8	8	security.

DKU7007 PRINTER

The printer must be defined by a separate TERMNL command declaring the "terminal-subtype" as PRT at CNC generation. Text for printing can only be sent to the printer address.

The three printing modes are treated as follows,

- transparent mode
- display image mode
- forms mode.

In forms and display image modes, printer control functions and "time-fills" are automatically generated by the controller.
In transparent mode, however, the user must ensure that the "time-fill" count is adequate for proper printer synchronization.

To define the "time-fill" count in transparent mode, proceed as follows,

- Determine the value to be given for the number of "time-fills", say, 6
  - Start from ASCII code 20 which is a "space", using the ASCII table
  - Count 6 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 25
  - The EBCDIC equivalent is 6C or graphic %
- In the MCS application, send a control sequence in one of the formats :

  ><US▽% using the graphic symbol

  ><US▽><C6C using the EBCDIC value in control character form

DKU7007
continued

DKU7007 PRINTER (continued)

Transparent Mode

Method of addressing is as follows,

- either address the printer with STA=3F in the VIP header
- or address the printer
  - with STA=00 in the VIP header
  - and, precede the text with ><ESCZ.

The message is printed as it is. The text in the message must contain all the necessary control codes specific to the printer and for formatting the text, see "Text for Display".

"Time-fill" control codes for the proper mechanical synchronization of the printer should also be included in the message text.

><USV defines the "time-fill" count, in the case where the printer is used for hard copy, and where the proper synchronization of the printer mechanism is required, see the previous page for programming the "time-fills".

Display Image Mode

Method of addressing is as follows,

- address the printer with STA=00 in the VIP header
- and, terminate the message text with ><ESCN.

The message text may optionally begin with ><ESCX or ><ESCY.

The text is formatted automatically on the printer according to the display characteristics of the format.

Printer control functions and "time-fills" are automatically generated by the controller.

Forms Mode

Method of addressing is as follows,

- address the printer with STA=00 in the VIP header
- and, terminate the message text with ><ESCM.

If the text begins with ><ESCX, only the variable fields will be printed.

If the text begins with ><ESCY, both fixed and variable fields will be printed.

If the text is not preceded by ><ESCX or ><ESCY, it will not be printed.

Variable fields defined as "inhibit-printing" will not be printed.

The controller automatically generates printer control functions and "time-fills".

DTU7171 operates on TTY line procedure

The display terminal unit is equipped with a keyboard and display screen.

The screen buffer size is organized as follows,

- 480 characters composed of 12 lines of 40 double sized characters
- 960 characters composed of 12 lines of 80 characters
- 1920 characters composed of 24 lines of 80 characters

Depending on an operator activated switch on the terminal, excess characters can either be overprinted on the last line or "rolled-up".

The character set contains 64 different graphic symbols or 95, if the lower case option is present. The types of graphic symbols are determined by the national options available.

A printer for hard copy can be attached to the terminal and is controlled by device function codes sent to the DTU7171 in output messages addressed to it.

DTU7171 DISPLAY

The information listed is a summary description and does not explain behavior near the screen boundaries.

The current entry position is marked by the position of the cursor.

><ACK position the cursor according to character position and line number

><BO3 next line, first character position

><BS▽ same line, one position to the left

><BEL acoustic signal to the operator

><CAN same line, next character position

><CR▽ same line, first character position

><DC2 turn on cursor and enable/resume printing

><DC4 turn off cursor and suppress printing

><EM▽ same as ><BS▽

><GS▽ first line, first character position, top left or "home"

><HT▽ same as ><BO3

><LF▽ next line, same character position

><NL▽ same as ><BO3

><RS▽ erase all characters from cursor position to the end of the line

><SUB previous line, same character position

><US▽ erase all characters from cursor position to the end of screen

><VT▽ same as ><LF▽

DTU7171
continued

To position the cursor, proceed as follows using the ASCII table

- Determine the value to be given for the character position, say, 37
  - Start from ASCII code 20 which is a "space"
  - Count 37 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 44
  - The EBCDIC equivalent is C4 or graphic D
- Determine the value to be given to the line number, say, 11
  - Use exactly the same method as above for character position
  - The EBCDIC equivalent is 5C or graphic *
- In the MCS application, send a control sequence in one of the formats :

>><ACKD* using graphic symbols

>><ACK><CC4><C5C using EBCDIC values in control character form

DTU7171 PRINTER ATTACHMENT

To resume printing, the cursor must be restored in one of two ways,

- by activating the CTR and R control keys, this action being done by the operator
- by sending the control code ><DC2, this action being done by the application.

The other control codes applicable to printer operations are,

><DC4 turn off cursor and suppress printing

><DLE start simultaneous printing and display

><ETB print contents of screen

IBM3270 operates on BSC3270 line procedure

The IBM3270 identifies the following components,

- 3271 Model 2 control unit in general poll mode
- 3277 Model 2 display station
- 3284 Model 2 printer.

The control unit has a 1920-character buffer capacity and can be configured with up to 32 display stations and printers.
At least 1 display station is needed for the control unit.

The display station comprises a screen with keyboard, capable of displaying 1920 characters per screen in 24 rows of 80 characters.
The screen can display the 64 standard ASCII characters.
The cursor indicates where the next character will appear on the screen and is represented by an underscore (_).

The printer has a 1920-character buffer, a 40-cps print rate and allows a choice of 120, 126 or 132 print positions per line.
The displayable characters on the screen can be reproduced as hard copy on the printer.

CNC DECLARATION

All display stations and printers configured to the same control unit are declared by individual TERMNL commands under one STATN command, as follows,

- each display station is declared as IBM3270 with the terminal-subtype KCT
- the printer associated with the display station immediately follows it and is declared as SLAVE with the terminal-subtype PRT
- the parameter ADD is mandatory for each TERMNL command declared, introducing values which are determined at the time of installation and are obtainable from the Field Engineering Service.

Associated with each TERMNL command is a QUEUE command in which the unedited mode is mandatorily declared as follows,

- for the terminal-subtype KCT, IM=UN, OM=UN
- for the terminal-subtype PRT, OM=UN.

In unedited mode, symbolic representation must not be used, see page 3-02.

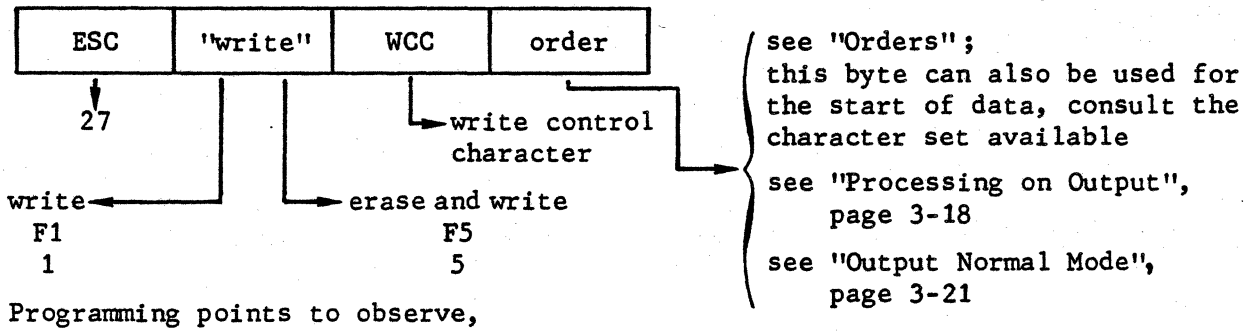
TDS ENVIRONMENT

The Forms utility, comprising FORMGEN and FORMRTP, is used to format the screen.

IBM3270
continued

COMMANDS : formats give EBCDIC values and, where applicable, the equivalent graphic symbol

- Write / Erase and Write : referred to in the text as "write" operations



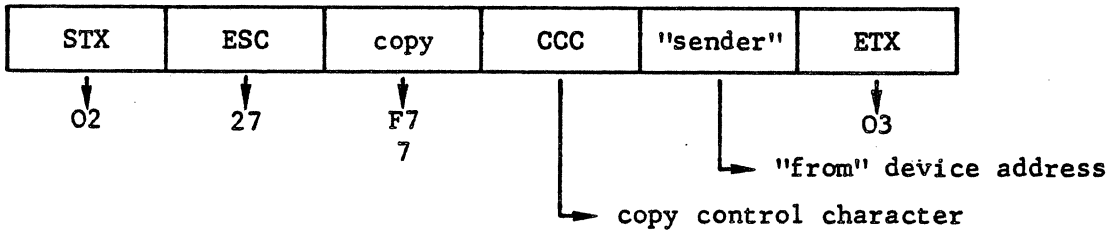
Programming points to observe,

- if commands are chained, "write" operations with the "start printer" bit set, must be the last in the chain
- the printout format bits are honored only if the "start printer" bit is set in the same WCC
- if a "write" operation includes data chained from a previous "write" operation, an SBA order must immediately follow the WCC to define the "start" address at which data entry is to commence.

Write Control Character for decoding, see "I/O Codes"	
Bit	Explanation
2-3	printout format, defined as follows, = 00 , NL and EM control codes in the data stream determine print line length; provides a 132-print position line when no control codes are present = 01 , specifies a 40-character print line = 10 , specifies a 64-character print line = 11 , specifies an 80-character print line
4	start printer : when set to 1, initiates a printout operation on completion of a "write" operation
5	sound alarm : when set to 1, sounds an audible alarm at the selected device at the end of an operation
6	keyboard restore : when set to 1, restores the functioning of the keyboard by resetting the INPUT INHIBITED on the screen; also resets the AID byte on termination of an I/O command
7	reset MDT : when set to 1, all MDTs in the data contained in the buffer of the selected device are reset before any further data is written or any orders executed

COMMANDS (continued)

- Copy



Programming points to observe,

- "Copy" should not be chained from a "write" operation
- If the "start printer" bit is set and commands are being chained, "copy" must be the last in the chain
- Once the data stream has been sent to the printer, the user program should send the following data stream to the display station that requested the copy, whereby the WCC in the data stream is to restore the keyboard :

STX	ESC	"write"	WCC	ETX

Copy Control Character for decoding, see "I/O Codes"	
Bit	Explanation
2-3	printout format, defined as follows, = 00 , NL and EM control codes in the data stream determine print line length; provides a 132-print position line when no control codes are present = 01 , specifies a 40-character print line = 10 , specifies a 64-character print line = 11 , specifies an 80-character print line
4	start printer : when set to 1, initiates a printout operation at the "receiver" printer once buffer transfers are completed
5	sound alarm : when set to 1, sounds an audible alarm at the selected "receiver" device once buffer transfers are completed
6-7	data type, defined for copying as follows, = 00 , only attributes are to be copied = 01 , attributes and unprotected alphanumeric fields are to be copied; protected fields are substituted for "null's" = 10 , attributes and protected alphanumeric fields are to be copied; unprotected fields are substituted for "null's" = 11 , entire contents of the storage buffer are to be copied

IBM3270
continued

COMMANDS (continued)

- Erase All Unprotected data

  EBCDIC 6F, Graphic Symbol ?

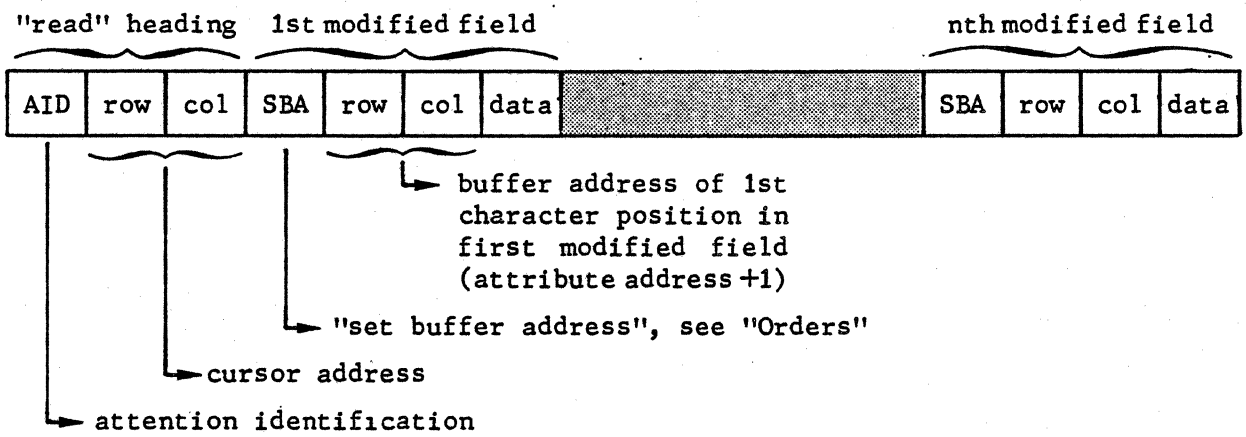
  The EAU command performs 5 functions at the addressed device,
  - clears all unprotected character locations to "null's"
  - resets the MDT bit for each unprotected field to zero, see WCC of "write"
  - unlocks the keyboard attached to the display
  - resets the AID byte, see AID of "read modified"
  - repositions the cursor to the 1st character location in the 1st unprotected field of the buffer; if no unprotected fields exist, the cursor is positioned to buffer location 0.

  The EAU command should not be chained to a "write" operation or to a "copy" command.

- Read Modified

  The "read modified" command is executed during the general polling sequence.

  The format of the "read" data stream is as follows,



  In the "read" heading, the AID byte, being the 1st byte in the data stream, identifies the function key activated at the keyboard by the operator. The user program can test the value of the AID code, and accordingly can perform the type of intervention required.

  Successive modified fields follow on after the "read" heading.
  As a field is modified by the operator, the MDT bit is set in the "attribute" byte for that field. Successive "attributes" are scanned for the set MDT bit, and when found, the data in the corresponding field is read, with "null's" suppressed, before the next "attribute" is examined.

  The SBA order code serves as a delimiter for the end of data of the preceding field and the start of the buffer address of the field following.

  If no fields have been modified, only the "read" heading will appear.

COMMANDS (continued)

- Read Modified (continued)

"Short" read denotes that no modified fields follow.

Attention Identification

Gp = group resultant transfer EB = EBCDIC value GrS = graphic symbol

Gp	Function Key(s)	EB	GrS	"read"
A	no AID generated (display station)	60	-	modified
	no AID generated (printer)	E8	Y	modified
B	ENTER and & selector pen attention	7D	'	modified
	PF 1	F1	1	modified
	PF 2	F2	2	modified
	PF 3	F3	3	modified
	PF 4	F4	4	modified
	PF 5	F5	5	modified
	PF 6	F6	6	modified
	PF 7	F7	7	modified
	PF 8	F8	8	modified
	PF 9	F9	9	modified
	PF 10	7A	:	modified
	PF 11	7B	#	modified
	PF 12	7C	@	modified
B	PF 13	C1	A	modified
	PF 14	C2	B	modified
	PF 15	C3	C	modified
	PF 16	C4	D	modified
	PF 17	C5	E	modified
	PF 18	C6	F	modified
	PF 19	C7	G	modified
	PF 20	C8	H	modified
	PF 21	C9	I	modified
	PF 22	4A	[¢	modified
	PF 23	4B	.	modified
	PF 24	4C	<	modified
C	selector pen attention space null	7E	=	modified
D	PA 1	6C	%	short
	PA 2 (CNCL)	6E	>	short
	PA 3	6B	,	short
	CLEAR	6D	_	short

A : field addresses and text in modified fields are transferred

B : the AID code, cursor address, SBA order, attribute address + 1 and text for each modified field are transferred;
"null's" are suppressed

C : the AID code, cursor address and field addresses are transferred;
no data is transferred

D : only the AID code is transferred

IBM3270
continued

DEVICE ADDRESSES

Device addresses used in the "copy" command are hardware configured and are determined at the time of installation.

These values, like those of the ADD parameter declared at network generation, are obtainable from the Field Engineering Service.

The "to" device identifies the "receiver" while the "from" device identifies the "sender".

I/O CODES

I/O codes represent the decoding of bits set by the user in the following parameters,

- WCC, write control character of "write" operations
- CCC, copy control character of the "copy" command
- "attribute" of the SF, start field order.

Bits 0 and 1 are omitted, since their values are not defined by the user but are instead reserved for use by the IBM3270.
Their settings do not affect the decoding of the byte to be specified.

I/O Codes

2 through 7 denote bits set by user; GrS = graphic symbol; EB = EBCDIC value

23 4567	GrS	EB	23 4567	GrS	EB	23 4567	GrS	EB	23 4567	GrS	EB
00 0000	∇	40	01 0000	&	50	10 0000	-	60	11 0000	0	F0
00 0001	A	C1	01 0001	J	D1	10 0001	/	61	11 0001	1	F1
00 0010	B	C2	01 0010	K	D2	10 0010	S	E2	11 0010	2	F2
00 0011	C	C3	01 0011	L	D3	10 0011	T	E3	11 0011	3	F3
00 0100	D	C4	01 0100	M	D4	10 0100	U	E4	11 0100	4	F4
00 0101	E	C5	01 0101	N	D5	10 0101	V	E5	11 0101	5	F5
00 0110	F	C6	01 0110	O	D6	10 0110	W	E6	11 0110	6	F6
00 0111	G	C7	01 0111	P	D7	10 0111	X	E7	11 0111	7	F7
00 1000	H	C8	01 1000	Q	D8	10 1000	Y	E8	11 1000	8	F8
00 1001	I	C9	01 1001	R	D9	10 1001	Z	E9	11 1001	9	F9
00 1010	[¢	4A	01 1010	] !	5A	10 1010	¦	6A	11 1010	:	7A
00 1011	.	4B	01 1011	$	5B	10 1011	,	6B	11 1011	#	7B
00 1100	<	4C	01 1100	*	5C	10 1100	%	6C	11 1100	@	7C
00 1101	(	4D	01 1101	)	5D	10 1101	_	6D	11 1101	'	7D
00 1110	+	4E	01 1110	;	5E	10 1110	>	6E	11 1110	=	7E
00 1111	! |	4F	01 1111	^ ¬	5F	10 1111	?	6F	11 1111	"	7F

ORDERS : formats give EBCDIC values

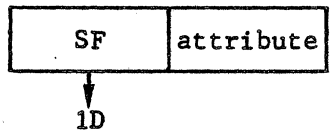
- IC - Insert Cursor :
  repositions the cursor to the location specified by the current buffer address.

  EBCDIC 13

- PT - Program Tabulation :
  advances the current buffer address to the address of the 1st character position of the next unprotected field.

  EBCDIC 05

- SF - Start Field :
  notifies the control unit that the next byte is an attribute in the "write" data stream to be stored at the current buffer address.

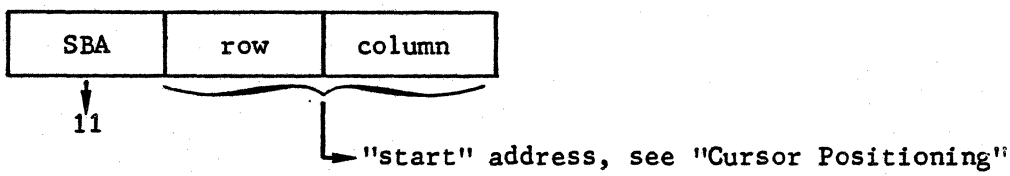


Start Field Attribute for decoding, see "I/O Codes"	
Bit	Explanation
2	0 = unprotected 1 = protected
3	0 = alphanumeric 1 = numeric only, causes automatic upper-case shift in data entry keyboard
4-5	00 = display / not selector-pen detectable 01 = display / selector-pen detectable 10 = intensified display / selector-pen detectable 11 = nondisplay, nonprint, not selector-pen detectable
6	must be zero
7	MDT (modified data tag) : identifies modified fields during "read modified" commands, as follows, 0 = field has not been modified 1 = field has been modified by the operator; can also be set by the application in the data stream.

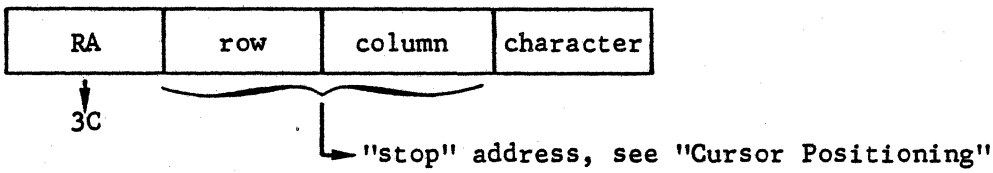
IBM3270
continued

ORDERS (continued)

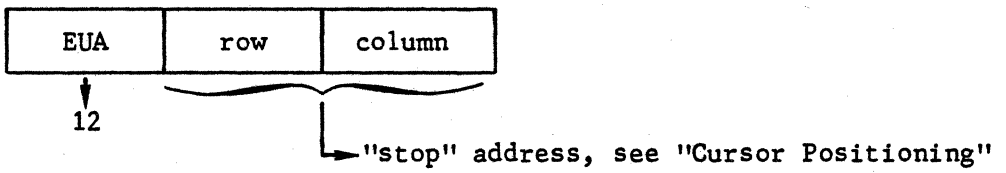
- SBA - Set Buffer Address :
  specifies a new buffer address from which "write" operations are to start or continue ; can precede all other orders in the data stream to indicate various areas of the buffer.



- RA - Repeat to Address :
  stores a specified character repetitively starting at the current buffer address and ending, but not including, the specified "stop" address.



- EUA - Erase Unprotected to Address :
  inserts "null's" in all unprotected buffer locations, starting at the current buffer address and ending, but not including, the specified "stop" address ; attributes remain unaffected.



To position the cursor, proceed as follows using the "Cursor Positioning" table opposite

- Determine the co-ordinates of the cursor, say row 7, column 64
- Row 7 has the EBCDIC value C7 only up to column 32 ; between columns 33 and 80, row 7 has the value C8.
  Since the column required is 64, the value for row 7 is therefore C8.
- Column 64 lies in the range between columns 59 and 66, with the corresponding consecutive EBCDIC values ranging from 5A through 61.

59	5A	60	5B	61	5C	62	5D	63	5E	64	5F	65	60	66	61

  By pairing the columns with their corresponding EBCDIC values, the value for column 64, shown unshaded, is 5F.
- The buffer address of the cursor at row 7, column 64 is "C8""5F".

Cursor Positioning
for 3277 Model 2 Display Station

R, C denote "row" and "column" respectively of the cursor, being the co-ordinates to reference the cursor.
R, C are the corresponding EBCDIC values for R, C respectively.

Intervening columns are omitted if their corresponding EBCDIC values are consecutive within a given range.
Rows are not repeated if their values remain unchanged from the initial value for a given range in the row.

R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C	R	C
01	01	40	40	03	50	C3	D1	06	27	C6	6A	09	01	4A	40	11	50	4D	D1	14	27	50	6A	17	01	D4	40	19	50	D7	D1	22	27	5A	6A
	02		C1		58		D9		32		6F		02		C1		58		D9		32		6F		02		C1		58		D9		32		6F
	10		C9		59		5A		33		FO		10		C9		59		5A		33		FO		10		C9		59		5A		33		FO
	11		4A		66		61		42		F9		11		4A		66		61		42		F9		11		4A		66		61		42		F9
	17		50		67		E2		43		7A		17		50		67		E2		43		7A		17		50		67		E2		43		7A
	18		D1		74		E9		48		7F		18		D1		74		E9		48		7F		18		D1		74		E9		48		7F
	26		D9		75		6A	06	49	C7	40		26		D9		75		6A	14	49	D1	40		26		D9		75		6A	22	49	5B	40
	27		5A		80		6F		50		C1		27		5A		80		6F		50		C1		27		5A		80		6F		50		C1
	34		61		58		C9		58		C9		34		61		58		C9		58		C9		34		61		58		C9		58		C9
	35		E2	04	01	C3	FO		59		4A		35		E2	12	01	4D	FO		59		4A		35		E2	20	01	D7	FO		59		4A
	42		E9		10		F9		65		5O		42		E9		10		F9		65		5O		42		E9		10		F9		65		5O
	43		6A		11		7A		66		D1		43		6A		11		7A		66		D1		43		6A		11		7A		66		D1
	48		6F		16		7F		74		D9		48		6F		16		7F		74		D9		48		6F		16		7F		74		D9
	49		FO	04	17	C4	40		75		5A		49		FO	12	17	4E	40		75		5A		49		FO	20	17	D8	40		75		5A
	58		F9		18		C1		80		5F		58		F9		18		C1		80		5F		58		F9		18		C1		80		5F
	59		7A		26		C9	07	01	C7	60		59		7A		26		C9	15	01	D1	60		59		7A		26		C9	23	01	5B	60
	64		7F		27		4A		02		61		64		7F		27		4A		02		61		64		7F		27		4A		02		61
01	65	C1	40		33		5O		03		E2	09	65	4B	40		33		5O		03		E2	17	65	D5	40		33		5O		03		E2
	66		C1		34		D1		10		E9		66		C1		34		D1		10		E9		66		C1		34		D1		10		E9
	74		C9		42		D9		11		6A		74		C9		42		D9		11		6A		74		C9		42		D9		11		6A
	75		4A		43		5A		16		6F		75		4A		43		5A		16		6F		75		4A		43		5A		16		6F
	80		4F		50		61		17		FO		80		4F		50		61		17		FO		80		4F		50		61		17		FO
					51		E2		26		F9	10	01	4B	50		51		E2		26		F9	18	01	D5	50		51		E2		26		F9
02	01	C1	50		58		E9		27		7A		02		D1		58		E9		27		7A		02		D1		58		E9		27		7A
	02		D1		59		6A		32		7F		10		D9		59		6A		32		7F		10		D9		59		6A		32		7F
	10		D9		64		6F	07	33	C8	40		11		5A		64		6F	15	33	D2	40		11		5A		64		6F	23	33	5C	40
	11		5A		65		FO		34		C1		18		61		65		FO		34		C1		18		61		65		FO		34		C1
	18		61		74		F9		42		C9		19		E2		74		F9		42		C9		19		E2		74		F9		42		C9
	19		E2		75		7A		43		4A		26		E9		75		7A		43		4A		26		E9		75		7A		43		4A
	26		E9		80		7F		49		50		27		6A		80		7F		49		50		27		6A		80		7F		49		50
	27		6A						50		D1		32		6F						50		D1		32		6F						50		D1
	32		6F	05	01	C5	40		58		D9		33		FO	13	01	4F	40		58		D9		33		FO	21	01	D9	40		58		D9
	33		FO		02		C1		59		5A		42		F9		02		C1		59		5A		42		F9		02		C1		59		5A
	42		F9		10		C9		66		61		43		7A		10		C9		66		61		43		7A		10		C9		66		61
	43		7A		11		4A		67		E2		48		7F		11		4A		67		E2		48		7F		11		4A		67		E2
	48		7F		17		50		74		E9	10	49	4C	40		17		50		74		E9	18	49	D6	40		17		50		74		E9
02	49	C2	40		18		D1		75		6A		50		C1		18		D1		75		6A		50		C1		18		D1		75		6A
	50		C1		26		D9		80		6F		58		C9		26		D9		80		6F		58		C9		26		D9		80		6F
	58		C9		27		5A						59		4A		27		5A						59		4A		27		5A				
	59		4A		34		61	08	01	C8	FO		65		5O		34		61	16	01	D2	FO		65		5O		34		61	24	01	5C	FO
	65		5O		35		E2		10		F9		66		D1		35		E2		10		F9		66		D1		35		E2		10		F9
	66		D1		42		E9		11		7A		67		D9		42		E9		11		7A		67		D9		42		E9		11		7A
	74		D9		43		6A		16		7F		74		D9		43		6A		16		7F		74		D9		43		6A		16		7F
	75		5A		48		6F	08	17	C9	40		75		5A		48		6F	16	17	D3	40		75		5A		48		6F	24	17	5D	40
	80		5F		49		FO		18		C1		80		5F		49		FO		18		C1		80		5F		49		FO		18		C1
					58		F9		26		C9	11	01	4C	60		58		F9		26		C9	19	01	D6	60		58		F9		26		C9
03	01	C2	60		59		7A		27		4A		02		61		59		7A		27		4A		02		61		59		7A		27		4A
	02		61						33		5O		03		E2		64		7F		33		5O		03		E2		64		7F		33		5O
	03		E2	05	64	C5	7F		34		D1		10		E9	13	65	50	40		34		D1		10		E9	21	65	5A	40		34		D1
	10		E9		65		40		42		D9		11		6A		66		C1		42		D9		11		6A		66		C1		42		D9
	11		6A		66		C1		43		5A		16		6F		74		C9		43		5A		16		6F		74		C9		43		5A
	16		6F		74		C9		50		61		17		FO		75		4A		50		61		17		FO		75		4A		50		61
	17		FO		75		4A		51		E2		26		F9		80		4F		51		E2		26		F9		80		4F		51		E2
	26		F9		80		4F		58		E9		27		7A	14	01	50	50		58		E9		27		7A	22	01	5A	50		58		E9
	27		7A	06	01	C6	50		59		6A		32		7F		02		D1		59		6A		32		7F		02		D1		59		6A
	32		7F		02		D1		64		6F	11	33	4D	40		10		D9		64		6F	19	33	D7	40		10		D9		64		6F
03	33	C3	40		10		D9		65		FO		34		C1		11		5A		65		FO		34		C1		11		5A		65		FO
	34		C1		11		5A		74		F9		42		C9		18		61		74		F9		42		C9		18		61		74		F9
	42		C9		18		61		75		7A		43		4A		19		E2		75		7A		43		4A		19		E2		75		7A
	43		4A		19		E2		80		7F		49		50		26		E9		80		7F		49		50		26		E9		80		7F
	49		50		26		E9																										

IBM3270
continued

TERMINAL SPECIFICS

Refer to the appropriate product manual.

KDS7255, 7275

KDS7255 and KDS7275 operate on VIP line Procedure

The KDS7255 identifies the KDS7255/7256, while the KDS7275 identifies both the KDS7265/7266 and the KDS7275/7276.

From the point of view of programming and operation, all versions of KDS configured on the DPS 7 are identical.

The KDS is a data-capture system operating in the following sequence,

- data is entered on the keyboard, and is checked and formatted locally
- it is then transcribed onto the diskette
- once the KDS is connected to the DPS 7, the data on the diskette is available for deferred transfer.

The visibility from the application is limited to a receive-only display and two diskette units.

KDS DISPLAY

The KDS display must be declared at network generation by a separate TERMNL command specifying the following mandatory parameters,

- CRT as the "terminal-subtype"
- ASSIGN, specifying either an input-queue as a "dummy" assignment or TDS
- AUTO, for logon to be performed by the communications software, since the KDS keyboard is not interactive.

The network control operator can modify the assignment during the communications session.

The screen is organized in 4 lines of 32 characters. Including the "space", 64 different graphic symbols can be displayed.

Display is limited to the first 128 characters of each message text.

Page overflow occurs when a message text exceeds 960 characters.

The only control code for managing the screen is ><FF∇ which serves to reset the screen before the message is displayed.

KDS7255, 7275
continued

KDS DISKETTE UNIT

The KDS diskette unit must be declared at network generation by a separate TERMNL command specifying DSK as the "terminal-subtype".

Messages with STA=00 in the VIP header can be exchanged between the application and the unit.

The characteristics of the data are,

- maximum size of the message is 960 characters, being the page size
- the size of the records on the diskette is determined by the diskette label, however, the presence of ><CR▽ or ><CR▽><LF▽ in a message sent to the diskette unit forces an end to the current record and data following it begins a new record.

The following control code sequences in mark form enable the operational handling of the diskette unit,

><EOF end-of-file, with no characters following to close the file

><ESCE><ESCR read data from diskette unit 1

><ESCF><ESCR read data from diskette unit 2

><ESCE><ESCS followed by data, write data to diskette unit 1

><ESCE><ESCS><FF▽><DELEOF end-of-file to be written to diskette unit 1

><ESCF><ESCS followed by data, write data to diskette unit 2

><ESCF><ESCS><FF▽><DELEOF end-of-file to be written to diskette unit 2

TERMINAL SPECIFICS

Refer to respective product manuals.

MTS7500/7508 operate on VIP line procedure

The MTS7500 and MTS7508 are multifunction terminal systems equipped with,

- a programmable processor with central memory
- a keyboard and a display screen with full ISO capability
- auxiliary terminals, such as,
  - diskette subsystem
  - printer.

The difference between the MTS7500 and the MTS7508 is that only the MTS7500 can be configured with a cassette unit comprising two cassette handlers.
The visibility of the multifunction terminal systems to the host is that of a VIP7700 with a printer and with

- either a diskette system, in the case of both MTS7500 and MTS7508
- or two cassette handlers, in the case of MTS7500.

MTS7500/7508 DISPLAY/KEYBOARD

The screen is organized in 12 lines of 80 characters. Including the "space", 95 different graphic symbols, comprising both upper and lower case, can be displayed.

A cursor is displayed as an aid in formatting the screen.

Local software controls the data format.

Header for Display/Keyboard

The 3 parameters in the VIP header have the following values,

- STA = EBCDIC 00 = COBOL Collating Sequence 1
- FC1 = last function code entered on the keyboard or echo to the application
- FC2 = last but one function code entered or echo to the application.

Text for Display

ENTRY MARKER :

The following terminal control codes in mark form alter the current entry position,

><B01 as for ><DC4, with display memory cleared, i.e., screen erased

><B03 equivalent to the sequence ><CR▽><LF▽

><BS▽ same line, 1 character position to the left

><CR▽ same line, leftmost character position

><DC1 one line up, same character position, i.e., reverse line feed

MTS7500, 7508
continued

ENTRY MARKER (continued) :

><DC2 same line, 1 character position to the right, i.e., forward space
><DC3 position cursor according to line number and character position
><DC4 uppermost line, leftmost character position, i.e., page return
><DEL follows ><HT▽, ><FF▽, ><B01 as a "time-fill"
><FF▽ same as ><B01
><HT▽ same line, first tabulation to the right
><LF▽ one line down, same character position
><NL▽ same as ><B03

To position the cursor, proceed as follows using the ASCII table

- Determine the value to be given for the line number, say, 11
  - Start from ASCII code 20 which is a "space"
  - Count 11 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 2A
  - The EBCDIC equivalent is 5C or graphic *
- Determine the value to be given for the character position, say, 37
  - Use exactly the same method as above for the line number
  - The EBCDIC equivalent is C4 or graphic D
- In the MCS application, send a control sequence in one of the formats :

  ><DC3*D using graphic symbols

  ><DC3><C5C><CC4 using EBCDIC values in control character form

PAGE OVERFLOW :

detection

a message OVERFLOW is displayed on the screen, if an attempt is made to send data greater than either 12 lines or 960 data characters.

cause

- result of a programming error
- result of an operator error in failing to clear the screen

correction

- press KEYBOARD control key and H simultaneously
- send the command $*$RDY
- if overflow occurs again as a result of message length, send the command $*$RDY STRONG to cancel the message.

The following functions for text for display are not supported, namely,

- blanking
- blinking
- forms mode
- tabulation

MTS7500, 7508
continued

MTS7500/7508 DISKETTE UNIT

The diskette unit must be defined by a separate TERMNL command declaring the "terminal-subtype" as DSK at CNC generation.

Messages with STA=00 in the VIP header can be exchanged between the application and the diskette unit.

The characteristics of the data are,

- maximum size of block is 960 characters, i.e., a maximum of 984 characters in 12 lines of 80 data characters plus the control codes CR▽LF▽
- maximum size of record is 249 characters, including the mandatory end-of-record control sequence ><CR▽><LF▽
- maximum size of last record in a block is 248 characters, including the mandatory end-of-block control sequence ><CR▽
- with the LOW SPEED option, data can only be received by the host processor as separate records
- with the HIGH SPEED option, only complete files can be read from or written to the diskette unit.

The control sequences for a diskette unit are,

><EOF end-of-file, with no characters following to close the file

><ESCR "read" command, issued once in the message to the diskette unit

><ESCS "write" command, followed by a block of records

To generate end-of-record control codes, do NOT use either COBOL facility,

- SEND AFTER ADVANCING clause
- BLOCKING option.

Either method will generate the control sequence ><CR▽><LF▽ in front of the record, thus making it an empty first record which is not acceptable to the MTS7500/7508.

When reading from the diskette unit, EOF denotes that the end-of-file has been reached.

When writing to the diskette unit, EOF followed by either ><CR▽><LF▽ or ><CR▽ is considered a normal record.

MTS7500 CASSETTE UNIT

The MTS7500 cassette unit must be defined by a separate TERMNL command declaring the "terminal-subtype" as CAS at CNC generation.

Messages with STA=00 in the VIP header can be exchanged between the application and the cassette unit.

The characteristics of the data for the MTS7500 cassette unit are the same as those which apply to the MTS7500 diskette unit.

Messages sent to the cassette unit must contain the following information, in the sequence as shown,

- selection of the cassette unit, whether front or rear
- type of command
- message text, where applicable.

SELECTION :

The following terminal control codes in mark form enable the selection of the cassette unit, in the sequence shown,

><ESCE selects the rear cassette handler , followed by ><ESCX

><ESCF selects the front cassette handler, followed by ><ESCX

><ESCX empty text

COMMAND :

The following terminal control codes in mark form enable the operation of the cassette handler selected,

><ESCP rewind to the start of cassette tape, followed by ><ESCX

><ESCR "read" command, issued once in the message to the cassette handler

><ESCS "write" command, followed by a block of records

The "backspace" function ><ESCD is not supported and must not be sent.

The MTS7500 disconnects the line when the control sequence ><ESCD is issued.

MTS7500, 7508
continued

MTS7500/7508 PRINTER

Printable text is defined between the control codes in mark form ><STX, to denote "start-of-text" and, either ><ETX or ><EM▽, to denote "end-of-text".

Text output to the printer does not appear on the screen.

° Method of addressing (1) :

- address the display/keyboard with STA=00 in the VIP header.

The keyboard is locked during printing.

The message is printed as it is. The text in the message must contain all necessary control codes for formatting the text, which are the same as for positioning the entry marker for display on the screen.

Like display on the screen, "time-fills" are required to ensure proper synchronization of the printer mechanism. Any control code which causes movement of the printer mechanism, such as "carriage-return" and "line-feed", must be immediately followed by a sequence of "time-fills", ><DEL, before any text is sent to be printed. Full line size capability is obtained by this method.

° Method of addressing (2) :

- define the printer by a separate TERMNL command declaring the "terminal-subtype as PRT at CNC generation
- address the display/keyboard with STA=00 in the VIP header.

The keyboard is not locked and may be used to enter data into the screen and to transmit the data while printing proceeds.

"Time-fills" may be needed to ensure proper synchronization of the printer.

Full line size capability is obtained by this method.

° Method of addressing (3) :

- define the printer by a separate TERMNL command declaring the "terminal-subtype" as PRT at CNC generation
- address the display/keyboard with STA=3F in the VIP header.

The keyboard is not locked and may be used to enter data into the screen and to transmit the data while printing proceeds.

The message is printed as it is. The text in the message must contain all necessary control codes for formatting the text, which are the same as for positioning the entry marker for display on the screen.

Like display on the screen, "time-fills" are required to ensure proper synchronization of the printer mechanism. Any control code which causes movement of the printer mechanism, such as "carriage-return" and "line-feed", must be immediately followed by a sequence of "time-fills", ><DEL, before any text is sent to be printed. Full line size capability is obtained by this method.

★

STS2840 operates on VIP line procedure

STS2840 is a generic name for a cluster of stations seen as a multipoint VIP line.

Each STS2840 station can contain up to 3 terminals, namely,

- a display with keyboard
- a printer
- a diskette unit.

STS2840 DISPLAY/KEYBOARD

The screen is organized in 16 or 24 lines of 80 character positions.

96 different graphic symbols can be displayed, their types depending on national options.

A marker line is displayed as an aid in formatting the screen.

Text for Display

The following terminal control codes in mark form enable the operational handling of the display,

><CR▽><LF▽ or ><LF▽, see ><ESCS

><ESCA resume expanded message format

><ESCC start companion hard copy on the printer

><ESCD request "send"

><ESCG acoustic signal to the operator and display "ALARME 31"

><ESCH move cursor one position to the left on the same line

><ESCI move cursor to the next tabulation stop on the right

><ESCJ erase end-of-line, and move cursor to the next line at first character position

><ESCK move cursor one line down at the same character position

><ESCL move cursor to the first line at first character position

><ESCN start of protected field

><ESCO start of unprotected field

><ESCQ move cursor one line up at the same character position

><ESCS move cursor to the next line at first character position

><ESCT access protected area

STS2840
continued

><ESCW erase end-of-line and any subsequent lines

><ESCX clear screen, and reset cursor in "home" position, i.e., at leftmost character position at the top of the screen

><ESCY erase field

><ESC) call "screen-format" from companion diskette, the 4 characters identifying the name of the "screen-format" must follow on immediately.

This control sequence must form a complete message in itself.

><ESC@ start compressed message format

><ESC\ display graphic symbol Ξ

><ESC] display one blank

><ESC^ subfield delimiter

><ESC_ move cursor one position to the right on the same line

><DC1 request screen dump

><DC2 position the cursor according to line number and character position

><DC4 position the cursor after display of the current message is completed

><FF▽ see ><ESCX

><GS▽ field delimiter, followed by an attribute character identifying the field

To position the cursor, proceed as follows using the ASCII table

- Determine the value to be given for the line number, say, 11
  - Start from ASCII code 20 which is a "space"
  - Count 11 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 2A
  - The EBCDIC equivalent is 5C or graphic *
- Determine the value to be given for the character position, say, 37
  - Use exactly the same method as above for the line number
  - The EBCDIC equivalent is C4 or graphic D
- In the MCS application, send a control sequence in one of the formats :

  ><DC2*D using graphic symbols

  ><DC2><C5C><CC4 using EBCDIC values in control character form

TN300/1200 operate on TTY line procedure

The TN300 and TN1200 are printers with the options,

- keyboard
- paper tape attachment
- cassette.

The keyboard can send any of the 128 ISO codes.

The character set for the printer comprises 94 graphic symbols (excluding the "space") according to the national options available.

Characters received with parity error are treated as follows,

- if TN300, an "interrupt" condition is set up at the terminal
- if TN1200, the erroneous character is printed with the graphic symbol ◇.

Line length depends on the model and the options available, namely,

- if TN300, 118 character positions
- if TN1200, either 80 or 120 character positions.

CONTROL CODES

Unless specifically stated, the control codes in mark form pertain to both TN300 and TN1200.

><BEL sounds an alarm

><BS▽ moves print one position to the left

><CR▽ resets print position to first tabulation stop or leftmost column

><DC1 starts paper tape reader forwards, optionally used for cassette

><DC2 starts paper tape punch, optionally used for cassette

><DC3 stops paper tape reader, optionally used for cassette

><DC4 stops paper tape punch, optionally used for cassette

><ESC0 starts paper tape reader backwards, optionally used for cassette

><ESC1 sets horizontal tabulation stop at current print position

><ESC2 clears all tabulation stops

><ESC3 applicable only for TN1200, selects red printing option

><ESC4 applicable only for TN1200, selects black printing option

><ESCH turns on printer motor

><ESCJ turns off printer motor

TN300, 1200
continued

CONTROL CODES (continued)

><ESCK turns on auxiliary device

><ESCL turns off auxiliary device

><ESC: resumes printing

><ESC; stops printing, allows transmission

><FF▽ applicable only for TN1200, throws paper to top-of-page
(option : fixed setting to 8, 8.5 or 11 inches)

><HT▽ moves print to next tabulation stop or end-of-line

><LF▽ advances paper depending on model,
- if TN300, one or two lines up (switch selectable)
- if TN1200, one line up (switch selectable, 3 or 6 lines per inch)

><VT▽ applicable only for TN1200, advances paper to next vertical tabulation stop or top-of-page

TERMINAL SPECIFICS

Refer to respective product manuals.

TTS7800 operates on VIP line procedure

TTS7800 DISPLAY/KEYBOARD

The screen is organized in 12 or 24 lines of 80 character positions.
The optional plasma screen is organized in 6 or 12 lines of 40 characters.

The keyboard has 64 keys and is organized a follows,

- an alphanumeric pad adaptable for national key layout options
- a numeric pad
- a selection of 13 fixed function-keys
- a selection of 8 variable function-keys.

Variable function-keys are used in conjunction with the fixed function-keys to select transactions via a menu.

Header for Display/Keyboard

The 3 parameters in the VIP header have the following values,

- STA = EBCDIC 00 = COBOL Collating Sequence 1
- FC1 = last function code entered on the keyboard or echo to the application
- FC2 = last but one function code entered or echo to the application.

Text for Display

BLANKING :

><CA1 displayed as a "space" or graphic ~ starts a character string which is blanked out on display.
A "space" or end-of-line ends the blanking.

BLINKING :

><BEL or ><BLK or graphic ¬ are displayed as graphic ^ and starts a blinking character string.
A "space" or end-of-line ends the blinking.

COBOL :

LINE and PAGE keywords have the same effect as ><B03 and ><B01 respectively, when used in a clause with the SEND verb.

TTS7800
continued

ENTRY MARKER :

The following terminal control codes in mark form alter the current entry position,

><BO1 as for ><DC4, with display memory cleared, i.e., screen erased

><BO3 equivalent to the sequence ><CR▽><LF▽

><BS▽ same line, 1 character position to the left

><CR▽ same line, leftmost character position

><DC1 one line up, same character position, i.e., reverse line feed

><DC2 same line, 1 character position to the right, i.e., forward space

><DC3 position cursor according to line number and character position

><DC4 uppermost line, leftmost character position, i.e., page return

><DEL follows ><HT▽, ><FF▽, ><BO1 as a "time-fill"

><FF▽ same as ><BO1

><HT▽ same line, first tabulation to the right

><LF▽ one line down, same character position

><NL▽ same as ><BO3

To position the cursor, proceed as follows using the ASCII table

- Determine the value to be given for the line number, say, 4
  - Start from ASCII code 20 which is a "space"
  - Count 4 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 23
  - The EBCDIC equivalent is 7B or graphic #
- Determine the value to be given for the character position, say, 15
  - Use exactly the same method as above for the line number
  - The EBCDIC equivalent is 4B or graphic .
- In the MCS application, send a control sequence in one of the formats :

  ><DC3#. using graphic symbols

  ><DC3><C7B><C4B using EBCDIC values in control character form

FORMS MODE :

The following terminal control codes in mark form set the terminal either in forms mode or in normal mode.
Forms are defined while in normal mode.

><ESCM enters the terminal into forms mode

><ESCN resets the terminal into normal mode

><FS▽ defines the start of a fixed field

><GS▽ defines the start of a variable field, followed by a value, as follows

0 the field contains a character to be transmitted and printed

1 inhibits printing

2 inhibits transmission

After the forms definition is completed, the terminal is entered into forms mode to allow protected operation under forms control.

MESSAGE BOUNDARIES :

In normal mode, message boundaries can be controlled by the application using the following control codes,

><ESCT start of message boundary

><ESCU end of message boundary.

TABULATION :

The following terminal control codes in mark form set the tabulation,

><ESC1 sets the tab stops at the current entry marker position

><ESC2 resets all the tab stops.

TTS7800
continued

TERMINAL SPECIFICS

Refer to the appropriate product manual.

TTU8124/8126 operate on TTY line procedure

The TTU8124 and TTU8126 terminals are printers with the optional features,

- keyboard
- front feed mechanism for inserting single single sheets
- paper loop for mechanically setting vertical tabulation and form length.

The character set for the printer consists of 63 graphic symbols, or 94, if the lower case option is present, according to the national options available.

The print line has a maximum capacity of 132 characters, or 80, as an option.

Characters received with a parity error are printed with the graphic symbol ◇.

When the last column position in a line is passed, an automatic new line movement occurs.
In the case of the TTU8126, the new line movement can be suppressed.

"Paper out" condition leads to disconnection for a switched line.

CONTROL CODES

The following terminal control codes in mark form enable the operational handling of the printer,

><BEL sounds an alarm

><BS▽ moves printer head one position to the left

><CR▽ resets printer head to leftmost column or first tabulation stop

><ESC0 followed by page length character, sets the page length for form feed

><ESC1 sets a horizontal tabulation at current print position
(10 positions to the inch; 16 tabulations stops, maximum)

><ESC2 clears all horizontal tabulations during operation
(horizontal tabs are automatically cleared at power-on)

><ESC3 sets a vertical tabulation at current line count
(6 lines to the inch; 10 tabulation stops, maximum)

><ESC4 clears all vertical tabulations during operation
(vertical tabs are automatically cleared at power-on)

><FS▽ ejects single sheets, automatically performed when less than 1 inch from bottom of the sheet

><GS▽ position single sheet to first print position, about half inch from top of the sheet

><HT▽ moves printer head to next horizontal tabulation or end-of-line

><LF▽ advances paper one line

><VT▽ advances paper to next vertical tabulation or top-of-page

TTU8124, 8126
continued

TERMINAL SPECIFICS

Refer to the respective product manuals.

TTU8221 operates on VIP line procedure

The TTU8221 is equipped with a keyboard and a printer with a buffer capacity of 960 characters.

95 different graphic symbols can be printed, basically the full ISO ASCII set, with variations depending on national keyboard options.

Its visibility to the host is that of a VIP7700 terminal with limited functions.

TTU8221 KEYBOARD/PRINTER

The page is organized according to parameters entered either by the operator or from the application.

Data entered on the keyboard is placed in a buffer, where editing can be carried out by the operator before the data is transferred to the host.

Printer and medium specifications are,

- a page is composed of from 10 to 90 lines
- a line contains from 38 to 132 characters, the default value being 132 characters
- paper width varies from 4" (10 cms) to 15" (38 cms)
- characters are 10 to the inch
- lines are 6 to the inch
- the minimum margin between the sprocket hole and the first character is ½", measured from center-to-center.

Header for Keyboard/Printer

The 3 parameters in the VIP header have the following values,

- STA = EBCDIC 3F = COBOL Collating Sequence 64,
  - when sent by the terminal
  - and, as forced by BTNS
- STA = EBCDIC 00 = COBOL Collating Sequence 1,
  - as received by the application
- FC1 = value as received by the application or overridden by the terminal operator
- FC2 = "▽", when the last character has been printed on fanfold paper
- FC2 = "!", when the last character has been printed on single forms.

TTU8221
continued

Control Codes

The following terminal control codes in mark form perform the mechanical functions of the printer,

><B01 new page, or form feed

><B03 new line

><CR∇><LF∇ see ><B03

><DC2 if preceding the following control codes in mark form, will perform a double paper movement for fanfold, namely,

- ><B01
- ><FF∇
- ><LF∇
- ><VT∇

><DC3 set page dimensions according to the number of lines and characters/line

><ESC1 set horizontal tabulation, up to 10 tab stops may be set in a line

><ESC2 clear all horizontal tabulation stops

><ESC5 set vertical tabulation, up to 10 tab stops may be set in a page

><ESC6 clear all vertical tabulation stops

><FF∇ see ><B03

><G2F sound a short acoustic signal

><HT∇ move print head to next horizontal tab stop, end-of-line is tab stop

><NL∇ see ><B03

><VT∇ skip paper to next horizontal tab stop, top-of-form is tab stop

To set the page dimensions, proceed as follows using the ASCII table

- Determine the value to be given for the number of lines, from 10 to 90, inclusive, say the number required is 55 lines
  - Subtract 9 from the number of lines required, that is (55 - 9) = 46
  - Start from ASCII code 20 which is a "space"
  - Count 46 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 4D
  - The EBCDIC equivalent is D4 or graphic M
- Determine the value to be given to the number of characters per line, from 38 to 132 inclusive, say the number required is 80 characters/line
  - Subtract 37 from the number of characters/line, i.e., (80 - 37) = 43
  - Use exactly the same method as above for the number of lines
  - The EBCDIC equivalent is D1 or graphic J
- In the MCS application, send a control sequence in one of the formats :

><DC3MJ using graphic symbols

><DC3><CD4><CD1 using EBCDIC values in control character form

TTU8221
continued

TTU8221 FRONT FEED

When the option is present, single sheets may be inserted from the front between the guides independent of the tractor mechanism.

Printer and medium specifications are,

- the vertical print area is 9 less than the total number of lines defined for the sheet
- the number of characters per line is set at 40, 48, 72 or 80.

The header for the normal traction printer as described previously, also applies to the "front-feed" printer.

Control Codes

The control codes listed for the normal traction printer as described previously also apply to the "front-feed" printer.

The additional control codes in mark form which apply specifically to the "front-feed" printer are,

><DC2 if preceding the following control codes in mark form, will perform a double paper movement for single sheet, namely,

- ><BO1
- ><FF▽
- ><LF▽
- ><VT▽

><ESC3 select single sheet for printing and move head to first print position on single sheet

><ESC4 select fanfold for printing and move head to first print position on fanfold.
This is selection by default at "power-on" and is only resorted to after the single sheet is ejected by "form-feed" or by excessive paper skips.

TTY33/35/37/38 operate on TTY line procedure

The TTY33, TTY35, TTY7 and TTY38 are printers with the options,

- keyboard
- paper tape attachment.

The keyboard can send either 96 or 128 ISO codes, depending on the model.

The character set for the printers comprises 63 graphic symbols, or 94 for the TTY37 and TTY38.

Line length is either 72 or 86 characters.

CONTROL CODES

The applicability of the control codes shown in mark form depends on the model and the installed options.

><BEL rings a bell
><BS▽ moves print position one character to the left
><CR▽ resets print position to leftmost margin
><DC1 starts paper tape reader
><DC2 starts paper tape punch
><DC3 stops paper tape reader
><DC4 stops paper tape punch
><ESC1 sets horizontal tabulation at current print position
><ESC2 clears all horizontal tabulations during operation *
><ESC3 selects printing in red
><ESC4 selects printing in black
><ESC5 sets vertical tabulation at line count
><ESC6 clears all vertical tabulations during operation *
><ESC7 rolls back paper one line
><ESC8 rolls back paper half a line
><ESC9 advances paper half a line
><FF▽ advances paper to top-of-page (next page)
><HT▽ moves print position to next tabulation stop on the same line
><LF▽ advances paper one line (3 or 6 lines to the inch)

* At power-on, all vertical and horizontal tabulations are automatically cleared.

TTY33, 35, 37, 38
continued

TERMINAL SPECIFICS

Refer to the respective product manuals.

★

VIP7001 operates on VIP line procedure

VIP7001 DISPLAY/KEYBOARD

The screen is organized in 12 or 24 lines of 80 characters. An option is available for 22 lines of 46 characters. Including the "space", 64 different graphic symbols, or 95 with lower case option, can be displayed. The types of graphic symbols depend on national options. An entry marker is displayed as an aid in formatting the screen.

The FORMGEN utility is available for formatting the screen. The FORMRTP utility, the Forms run-time package, is then executed.

Header for Display/Keyboard

The 3 parameters in the VIP header have the following values,

- STA = EBCDIC 00 = COBOL Collating Sequence 1
- FC1 = last function code entered on the keyboard or echo to the application
- FC2 = last but one function code entered or echo to the application.

Text for Display

BLANKING :

><CA1 displayed as a "space" or graphic ~ starts a character string which is blanked out on display.
A "space" or end-of-line ends the blanking.

BLINKING :

><BEL or ><BLK or graphic ¬ are displayed as graphic ^ and starts a blinking character string.
A "space" or end-of-line ends the blinking.

COBOL :

LINE and PAGE keywords have the same effect as ><B03 and ><B01 respectively, when used in a clause with the SEND verb.

VIP7001
continued

ENTRY MARKER :

The following terminal control codes in mark form alter the current entry position,

><B01 as for ><DC4, with display memory cleared, i.e., screen erased

><B03 equivalent to the sequence ><CR▽><LF▽

><BS▽ same line, 1 character position to the left

><CR▽ same line, leftmost character position

><DC1 one line up, same character position, i.e., reverse line feed

><DC2 same line, 1 character position to the right, i.e., forward space

><DC3 position cursor according to line number and character position

><DC4 uppermost line, leftmost character position, i.e., page return

><DEL follows ><HT▽, ><FF▽, ><B01 as a "time-fill"

><FF▽ same as ><B01

><HT▽ same line, first tabulation to the right

><LF▽ one line down, same character position

><NL▽ same as ><B03

To position the cursor, proceed as follows using the ASCII table

- Determine the value to be given for the line number, say 12
  - Start from ASCII code 20 which is a "space"
  - Count 12 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 2B
  - The EBCDIC equivalent is 4E or graphic +
- Determine the value to be given for the character position, say, 6
  - Use exactly the same method as above for the line number
  - The EBCDIC equivalent is 6C or graphic %
- In the MCS application, send a control sequence in one of the formats :

><DC3+% using graphic symbols

><DC3><C4E><C6C using EBCDIC values in control character form

FORMS MODE :

The following terminal control codes in mark form set the terminal either in forms mode or in normal mode.

Forms are defined while in normal mode.

><ESCM enters the terminal into forms mode

><ESCN resets the terminal into normal mode

><FS∇ defines the start of a fixed field

><GS∇ defines the start of a variable field, followed by a value, as follows

0 the field contains a character to be transmitted and printed

1 inhibits printing

2 inhibits transmission

After the forms definition is completed, the terminal is entered into forms mode to allow protected operation under forms control.

MESSAGE BOUNDARIES :

In normal mode, message boundaries can be controlled by the application using the following control codes,

><ESCT start of message boundary

><ESCU end of message boundary

PAGE OVERFLOW :

detection
- ERROR indicator on the terminal lights up.

cause
- result of a programming error
- result of operator error in failing to clear the screen.

correction
- press CTR and CLEAR control keys simultaneously
- send the command $*$RDY
- if overflow recurs, send $*$RDY STRONG to cancel the message.

TABULATION :

The following terminal control codes in mark form set the tabulation,

><ESC1 sets the tab stop at the current marker position

><ESC2 resets all the tab stops.

VIP7001
continued

VIP7001 PRINTER

Printable text is defined between the control codes in mark form ><STX, to denote "start-of-text" and, either ><ETX or ><EM▽, to denote "end-of-text". Text output to the printer does not appear on the screen.

° Method of addressing (1) :

- address the display/keyboard with STA=3F in the VIP header.

The keyboard is locked during printing.

The message is printed as it is. The text in the message must contain all necessary control codes for formatting the text, which are the same as for positioning the entry marker for display on the screen.

Like display on the screen, "time-fills" are required to ensure proper synchronization of the printer mechanism. Any control code which causes movement of the printer mechanism, such as "carriage-return" and "line-feed", must be immediately followed by a sequence of "time-fills", ><DEL, before any text is sent to be printed. Full line size capability is obtained by this method.

° Method of addressing (2) :

- define the printer by a separate TERMNL command declaring the "terminal-subtype" as PRT at CNC generation
- address the display/keyboard with STA=00 in the VIP header.

The keyboard is not locked and may be used to enter data into the screen and to transmit the data while printing proceeds.

Data is printed according to the standard VIP7001 hard copy format with a maximum line length of 80 characters. Print format controls, such as "carriage-return" and "line-feed", are generated automatically by the terminal controller.

° Method of addressing (3) :

- define the printer by a separate TERMNL command declaring the "terminal-subtype" as PRT at CNC generation
- address the display/keyboard with STA=3F in the VIP header.

The keyboard is not locked and may be used to enter data into the screen and to transmit the data while printing proceeds.

The message is printed as it is. The text in the message must contain all necessary control codes for formatting the text, which are the same as for positioning the entry marker for display on the screen.

Like display on the screen, "time-fills" are required to ensure proper synchronization of the printer mechanism. Any control code which causes movement of the printer mechanism, such as "carriage-return" and "line-feed", must be immediately followed by a sequence of "time-fills", ><DEL, before any text is sent to be printed. Full line size capability is obtained by this method.

VIP7100 operates on TTY line procedure

The VIP7100 is a keyboard/display terminal.

Entry of data in the display starts at the bottom line and on completion, the line is "rolled up" if the new line function has been provided for. Otherwise, excess characters overprint the rightmost character position, that is, position 80.

The "home" position defines the leftmost character position of the bottom line.

A cursor is displayed to facilitate data entry.

Screen capacity is 12 lines of 80 characters, with an option available for 24 lines.

The character set for the display consists of 63 graphic symbols, or 94, if the lower case option is present.

The keyboard can send any of the 128 ISO codes, however, if only the upper case is present, the set is reduced to 94.

CONTROL CODES

The following terminal control codes in mark form enable the operational handling of the display,

><BEL sounds an alarm

><BS▽ moves cursor one position to the left

><CR▽ resets cursor at the left margin, without affecting characters already displayed on the line

><DC2 moves cursor one position to the right

><FF▽ resets cursor to the left margin for erasing the screen

><LF▽ "rolls up" lines by one line in order to allow data entry into the bottom line

VIP7100
continued

TERMINAL SPECIFICS

Refer to the appropriate product manual.

VIP7200 operates on TTY line procedure

The VIP7200 is a keyboard/display terminal.

A cursor is displayed to indicate the current entry position. The screen is filled from the top, progressing line by line to the bottom. When the last line at the bottom is filled, the contents of the screen are automatically "rolled up" by one line in order to allow further data entry.

Screen capacity is 24 lines of 80 characters.

The character set for the display consists of 63 graphic symbols, or 94, if the lower case option is present.

The keyboard can send any of the 128 ISO codes.

CONTROL CODES

The following terminal control codes in mark form enable the operational handling of the display,

><BEL acoustic signal to the operator

><CR▽ reset cursor at left margin on the same line

><ESC3 set normal intensity for screen illumination

><ESC4 set low intensity for screen illumination

><ESCA move cursor one line up but at the same character position

><ESCB move cursor one line down but at the same character position

><ESCC move cursor one character position to the right on the same line

><ESCD move cursor one character position to the left on the same line

><ESCH reset cursor in "home" position, i.e., at leftmost character position at the top of the screen

><ESCJ erase text display from current cursor position to the end-of-page

><ESCK erase text display from current position to the end-of-line

><ESC' reset terminal to initial state, as follows,

- erase the screen
- set cursor to "home" position
- set normal intensity for screen illumination.

The equivalent control sequence is ><ESC><C79.

><ESCf position the cursor according to character position and line number

><ESCi send data from the start-of-line or start-of-page, depending on switch setting, to the current cursor position

VIP7200
continued

CONTROL CODES (continued)

><ESCn terminal to indicate its cursor position according to the format given in ><ESCf

><LF▽ move cursor one line down; if the cursor is on the bottom line, "roll up" will occur

To position the cursor, proceed as follows using the ASCII table

. Determine the value to be given for the character position, say, 37
  - Start from ASCII code 20 which is a "space"
  - Count 37 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 44
  - The EBCDIC equivalent is C4 or graphic D
. Determine the value to be given to the line number, say, 11
  - Use exactly the same method as above for character position
  - The EBCDIC equivalent is 5C or graphic *
. In the MCS application, send a control sequence in one of the formats :

><ESCfD* using graphic symbols

><ESCf><CC4><C5C using EBCDIC values in control character form

VIP7700 operates on VIP line procedure

VIP7700 DISPLAY/KEYBOARD

The screen is organized in 12 or 24 lines of 80 characters. An option is available for 22 lines of 46 characters. Including the "space", 64 different graphic symbols, or 95 with lower case option, can be displayed. The types of graphic symbols depend on national options. An entry marker is displayed as an aid in formatting the screen.

The FORMGEN utility is available for formatting the screen. The FORMRTP utility, the Forms run-time package, is then executed.

Header for Display/Keyboard

The 3 parameters in the VIP header have the following values,

- STA = EBCDIC 00 = COBOL Collating Sequence 1
- FC1 = last function code entered on the keyboard or echo to the application
- FC2 = last but one function code entered or echo to the application.

Text for Display

BLANKING :

><CA1 displayed as a "space" or graphic ~ starts a character string which is blanked out on display.
A "space" or end-of-line ends the blanking.

BLINKING :

><BEL or ><BLK or graphic ¬ are displayed as graphic ^ and starts a blinking character string.
A "space" or end-of-line ends the blinking.

COBOL :

LINE and PAGE keywords have the same effect as ><B03 and ><B01 respectively, when used in a clause with the SEND verb.

VIP7700
continued

ENTRY MARKER :

The following terminal control codes in mark form alter the current entry position,

><BO1 as for ><DC4, with display memory cleared, i.e., screen erased

><BO3 equivalent to the sequence ><CR▽><LF▽

><BS▽ same line, 1 character position to the left

><CR▽ same line, leftmost character position

><DC1 one line up, same character position, i.e., reverse line feed

><DC2 same line, 1 character position to the right, i.e., forward space

><DC3 position cursor according to line number and character position

><DC4 uppermost line, leftmost character position, i.e., page return

><DEL follows ><HT▽, ><FF▽, ><BO1 as a "time-fill"

><FF▽ same as ><BO1

><HT▽ same line, first tabulation to the right

><LF▽ one line down, same character position

><NL▽ same as ><BO3

To position the cursor, proceed as follows using the ASCII table

- Determine the value to be given for the line number, say, 11
  - Start from ASCII code 20 which is a "space"
  - Count 11 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 2A
  - The EBCDIC equivalent is 5C or graphic *
- Determine the value to be given for the character position, say, 37
  - Use exactly the same method as above for the line number
  - The EBCDIC equivalent is C4 or graphic D
- In the MCS application, send a control sequence in one of the formats :

><DC3*D using graphic symbols

><DC3><C5C><CC4 using EBCDIC values in control character form

FORMS MODE :

The following terminal control codes in mark form set the terminal either in forms mode or in normal mode.
Forms are defined while in normal mode.

><ESCM enters the terminal into forms mode

><ESCN resets the terminal into normal mode

><FS∇ defines the start of a fixed field

><GS∇ defines the start of a variable field, followed by a value, as follows

0 the field contains a character to be transmitted and printed

1 inhibits printing

2 inhibits transmission

After the forms definition is completed, the terminal is entered into forms mode to allow protected operation under forms control.

MESSAGE BOUNDARIES :

In normal mode, message boundaries can be controlled by the application using the following control codes,

><ESCT start of message boundary

><ESCU end of message boundary

PAGE OVERFLOW :

detection : ERROR indicator on the terminal lights up.

cause : Either one of the following conditions has occurred
- a result of a programming error
- a result of an operator error in failing to clear the screen

correction : Perform the following sequence of operations
- press CTR and CLEAR control keys simultaneously
- send the command $*$RDY
- if overflow occurs again as a result of message length, send the command $*$RDY STRONG to cancel the message.

TABULATION :

The following terminal control codes in mark form set the tabulation,

><ESC1 sets the tab stop at the current entry marker position

><ESC2 resets all the tab stops.

★

VIP7700
continued

★

VIP7700 PRINTER

Printable text is defined between the control codes in mark form ><STX, to denote "start-of-text" and, either ><ETX or ><EMV, to denote "end-of-text".

Text output to the printer does not appear on the screen.

° Method of addressing (1) :

- address the display/keyboard with STA = 3F in the VIP header.

The keyboard is locked during printing.

The message is printed as it is. The text in the message must contain all necessary control codes for formatting the text, which are the same as for positioning the entry marker for display on the screen.

Like display on the screen, "time-fills" are required to ensure proper synchronization of the printer mechanism. Any control code which causes movement of the printer mechanism, such as "carriage-return" and "line-feed", must be immediately followed by a sequence of "time-fills", ><DEL, before any text is sent to be printed. Full line size capability is obtained by this method.

° Method of addressing (2) :

- define the printer by a separate TERMNL command declaring the "terminal-subtype" as PRT at CNC generation
- address the display/keyboard with STA = 00 in the VIP header.

The keyboard is not locked and may be used to enter data into the screen and to transmit the data while printing proceeds.

Data is printed according to the standard VIP7700 hard copy format with a maximum line length of 80 characters. Print format controls, such as "carriage-return" and "line-feed", are generated automatically by the terminal controller.

° Method of addressing (3) :

- define the printer by a separate TERMNL command declaring the "terminal-subtype" as PRT at CNC generation
- address the display/keyboard with STA = 3F in the VIP header.

The keyboard is not locked and may be used to enter data into the screen and to transmit the data while printing proceeds.

The message is printed as it is. The text in the message must contain all necessary control codes for formatting the text, which are the same as for positioning the entry marker for display on the screen.

Like display on the screen, "time-fills" are required to ensure proper synchronization of the printer mechanism. Any control code which causes movement of the printer mechanism, such as "carriage-return" and "line-feed", must be immediately followed by a sequence of "time-fills", ><DEL, before any text is sent to be printed. Full line size capability is obtained by this method.

VIP7760 operates on VIP line procedure

VIP7760 DISPLAY/KEYBOARD

The screen is organized in 12 or 24 lines of 80 characters. Including the "space" 95 different graphic symbols, basically ISO/ASCII set with national options, can be displayed. An entry marker is displayed as an aid in formatting the screen.

The FORMGEN utility is available for formatting the screen. The FORMRTP utility, the Forms run-time package, is then executed.

Text for Display

The control sequences for tabulation, entry marker control and message boundaries are the same as for the VIP7700. Blinking and blanking are supported as options.

FORMS MODE

The following terminal control codes in mark form set the terminal either in forms mode or in normal mode. Forms are defined when in normal mode.

><ESCM enters the terminal into forms mode

><ESCN resets the terminal into normal mode

><FS▽ defines the start of a fixed field

><GS▽ defines the start of a variable field, followed by a value, as follows

- 0 the field is right-justified, non-repeated and alphanumeric, to be transmitted and printed
- 1 inhibits printing
- 2 inhibits transmission
- 4 numeric-only field
- 8 indicates a number of a line field to be repeated
- 16 the field is left-justified

><RS▽ terminates a repetitive line field, followed by a repetition code

value of repetition code : add 64 to the number of times that the line field is to be repeated.

VIP7760
continued

VIP7760 PRINTER

The printer must be defined by a separate TERMNL command declaring the "terminal-subtype" as PRT at CNC generation. Text for printing can only be sent to the printer address. The three printing modes are treated as follows,

TRANSPARENT MODE

Method of addressing : Either
- address the printer with STA=3F in the VIP header

Or
- address the printer with STA=00 in the VIP header
- and, precede the text with ><ESCZ.

The message is printed as it is. The text in the message must contain any necessary format control codes, see "Text for Display", and any other control codes specific to the printer.

"Time-fill" control codes for the proper mechanical synchronization of the printer should also be included in the message text.

><USV defines the "time-fill" count, in the case where the printer is used for hard copy, and where the proper synchronization of the printer mechanism is required.

To define the "time-fill" count, proceed as follows using the ASCII table

- Determine the value to be given for the number of "time-fills", say, 4
  - Start from ASCII code 20 which is a "space"
  - Count 4 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 23
  - The EBCDIC equivalent is 7B or graphic #
- In the MCS application, send a control sequence in one of the formats :

  ><USV# using the graphic symbol

  ><USV><C7B using the EBCDIC value in control character form

VIP7760 PRINTER (continued)

DISPLAY IMAGE MODE :

Method of addressing :

- address the printer with STA=00 in the VIP header
- and, end the message text with ><ESCN.

The message text may optionally begin with ><ESCX or ><ESCY.

The text is formatted automatically on the printer according to the display format characteristics.

Printer control functions and "time-fills" are generated automatically by the VIP7760 system.

FORMS MODE :

Method of addressing :

- address the printer with STA=00 in the VIP header
- and, end the message text with ><ESCM.

If the message text begins with ><ESCX, only the variable fields will be printed.

If the message text begins with ><ESCY, both the fixed fields and the variable fields will be printed.

If the message text is not preceded by either ><ESCX or ><ESCY, it will not be printed.

Variable fields defined as "inhibit-printing" will not be printed.

The printer is controlled automatically by the VIP7760 system.

VIP7760 DISKETTE

The diskette unit must be defined by a separate TERMNL command declaring the "terminal-subtype" as DSK at CNC generation.

Messages with STA=00 in the VIP header can be exchanged between the application and the unit, where the appropriate control sequence indicates that the message that would otherwise appear on the screen is destined for the diskette unit.

The diskette is used for forms management in the following ways,

- a form is stored by transmitting the form data, for formatting the screen, together with the necessary "dataset" name and form label, and the "diskette-write" control sequence, ><ESCW
- during a communications session, a form is retrieved, for dynamically formatting the screen, by transmitting the "dataset" name and form label, and the "diskette-read" control sequence, ><ESCV
- a form is deleted by transmitting the "dataset" name and the form label, and the "diskette-erase" control sequence, ><ESCQ.

The diskette control sequences must be placed after ><STX in the text portion of the message, but are not initiated until ><ETX received.

Control Code Sequences

In the following explanation of control sequences, it is assumed that

- the diskette is operational
- the "dataset" name is valid and has been entered before the control sequence
- the form name is not identical to one listed in the catalog.

The control sequences are listed in order of their functions,

><ESCW initiates the storage of a form under the specified "dataset" name and form label

><ESCV retrieves a local form and initiates the transmission of the form contents to the host, under the specified "dataset" name and form label

For both the ><ESCV and ><ESCW control sequences, the keyboard remains locked.

><ESCQ erases a single form or forms from a "dataset" as follows,
- a single form is erased, if both the "dataset" name and form label are specified
- forms are deleted from the "dataset", if the "dataset" name is specified, and then followed by the keyword $ALL

><ESCB enables the local display without transmit of a form under the specified "dataset" name and form label.

VIP7802 operates on TTY line procedure

The VIP7802 identifies the VIP7801 and VIP7802.

The screen is formatted in 24 lines of 80 characters, plus a 25th 80-character status line. This status line serves as a constant indicator of the terminal operating conditions, such as, terminal mode, terminal status and cursor position.

The keyboard is capable of producing 139 displayable characters, of which,

- 95 are standard ASCII characters, including the "space" and the lower-case option
- 33 are graphic symbols used in "communications display" mode
- 11 are line graphic symbols for line drawings, histograms and form outlines.

The position of the character to be displayed is indicated by the current position of the cursor which can be selected either as an "underscore" (_) or as a block.

The VIP7801/7802 transmits in three modes,

- character mode, in which information is transmitted as it is being keyed in, character at a time
- text mode, in which the terminal transmits from either the 1st character entered since the last transmission or a pre-defined position set by the Level 64, to the current cursor position
- forms mode, in which data to be transmitted is determined by the forms attributes qualifying each field.

ATTRIBUTES

The following control code sequences in mark form indicate the specific attribute to be stored at the current cursor position.

><ESC s	"visual", screen display in normal intensity
><ESC s{A|a}	"forms", restricts entry to alpha-characters and normal punctuation marks for editing
><ESC s{B|b}	"visual", data and underscore to blink, inverse video is not affected
><ESC s{D|d}	"forms", restricts entry to decimal digits and "space" negates ><ESC s{N|n}
><ESC s{E|e}	"forms", entry required before tabbing from the field
><ESC s{F|f}	"forms", field must be completely filled before tabbing
><ESC s{H|h}	"visual", data entered is not to be displayed, inverse video and underscore are not affected

VIP7802
continued

ATTRIBUTES (continued)

><ESC s{I|i} "visual", inverse video

><ESC s{J|j} "forms", field entry to be right justified

><ESC s{L|l} "visual", screen display in low intensity

><ESC s{M|m} "forms", field to be transmitted only if its contents is changed
negates ><ESC s{T|t} , sets ><ESC s{U|u}

><ESC s{N|n} "forms", restricts field to signed decimals and value indicators
negates ><ESC s{D|d}

><ESC s{P|p} "forms", defines protected field without changing other attributes
modifies ><ESC s{M|m} and ><ESC s{U|u}

><ESC s{R|r} "restore", negates attributes set for a field without altering the field assignment

><ESC s{T|t} "forms", transmits the contents of a protected field

><ESC s{U|u} "forms", defines unprotected field whereby data is to be systematically transmitted and made accessible to the operator

><ESC s _ "forms", display underscore

To set up a form, proceed as follows,
- send ><ESC ' ><ESC [h to clear the screen and set terminal in forms mode
- position cursor at start of 1st field to enter "attribute", then contents
- repeat same procedure for all fields until the form is completed.

To delete any attribute, send the control code sequence ><ESC [Q

KEYBOARD

><ESC [X locks keyboard, disables all keyboard functions except for RIS, RES, AUTO-LF, LOCAL and BREAK

><ESC [W unlocks keyboard, negates ><ESC [X

TABULATION : also see "Cursor"

><ESC p sets tab stop at current cursor position; invalid in forms mode

><ESC [g clears tab stop, negates ><ESC p ; invalid in forms mode

><ESC [N clears all tabs irrespective of cursor position; invalid in forms mode

><ESC [Z moves cursor backwards to the previously defined tab position

><HT▽ moves cursor forwards to next character position at which the tab was set; if in forms mode, cursor moves to start of next unprotected field

CURSOR : also see "Tabulation"

><BS▽ moves cursor 1 character position backwards on same line, regardless of fields transversed; cursor does not move out of position 1

><CR▽ reset cursor at left margin, character position 1, on same line

><ESC A moves cursor upwards by 1 line, same character position; when in line 1, cursor moves to line 24

><ESC B moves cursor downwards by 1 line, same character position; when in line 24, cursor moves to line 1

><ESC C moves cursor forwards by 1 character position along same line; when out of position 80 of line 24, cursor moves to "home" position

><ESC D moves cursor backwards by 1 character position along same line; when out of "home" position, cursor moves to position 80 of line 24

><ESC f positions cursor according to character position and line number

><ESC H moves cursor to the "home" position

><ESC n terminal to indicate its cursor position in the status line, according to the format given in ><ESC f

><LF▽ moves cursor downwards by 1 line; if cursor is already in line 24,
- in "roll" mode, data is rolled up by 1 line
- in "non-roll" mode, cursor does not move and "data-overflow" occurs

To position the cursor, proceed as follows using the ASCII table

- Determine the value to be given for the character position, say, 37
  - Start from ASCII code 20 which is a "space"
  - Count 37 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 44
  - The EBCDIC equivalent is C4 or graphic D
- Determine the value to be given to the line number, say, 11
  - Use exactly the same method as above for character position
  - The EBCDIC equivalent is 5C or graphic *
- In the MCS application, send a control sequence in one of the formats :

  ><ESCfD* using graphic symbols

  ><ESCf><CC4><C5C using EBCDIC values in control character form

VIP7802
continued

EDITING

><ESC J erases to end of page starting from current cursor position, tabs and attributes remaining unaffected; in forms mode, only unprotected data is erased

><ESC K erases to end of field starting from current cursor position; not valid if cursor is already in character position 80

><ESC [I allows characters to be inserted at current cursor position; data surpassing line width is truncated

><ESC [J negates ><ESC [I

><ESC [L inserts single blank line at line indicated by cursor; invalid in forms mode or when cursor is in the status line

><ESC [M deletes line in which cursor is located, including all attributes

><ESC [P deletes character within a line whereby all text is closed up to the left, tabs and attributes remaining unaffected

MODE

><ESC F resets the line graphics mode, negates ><ESC G

><ESC G sets terminal in line graphics mode, in which any of the 11 line graphic symbols can be used

><ESC I causes terminal to transmit next block when in block transmission mode

><ESC k sets terminal in character mode

><ESC q resets the roll mode, negates ><ESC r

><ESC r sets terminal in roll mode unless terminal is in forms mode

><ESC [E resets the block transmission mode

><ESC [block1 block2 block3 block4 F
sets terminal in block transmission mode according to the block-size designated; invalid if terminal is in character mode

><ESC [h sets terminal in forms mode, whereby data can only be entered in unprotected fields

><ESC [l sets terminal in text mode, whereby data can be entered anywhere

><ESC ` resets screen to initial state as follows,
- erase the screen
- set cursor to "home" position
- set text mode
- clear all attributes
- set normal intensity for screen illumination

VIP7804 operates on VIP line procedure

The VIP7804 identifies the VIP7804 and VIP7805.

The screen is formatted in 24 lines of 80 characters, plus a 25th 80-character status line. This status line serves as a constant indicator of the terminal operating conditions, such as, terminal mode, terminal status and cursor position.

The keyboard is capable of producing 139 displayable characters, of which,

- 95 are standard ASCII characters, including the "space" and the lower-case option
- 33 are graphic symbols used in "communications display" mode
- 11 are line graphic symbols for line drawings, histograms and form outlines.

The position of the character to be displayed is indicated by the current position of the cursor which can be selected either as an "underscore" (_) or as a block.

The VIP7804/7805 transmits in two modes,

- text mode, in which the terminal transmits from either the 1st character entered since the last transmission or a pre-defined position set by the Level 64, to the current cursor position
- forms mode, in which data to be transmitted is determined by the forms attributes qualifying each field.

Data buffering in the printer adapter allows printing to take place simultaneously with screen display. The adapter provides maximum buffering for up to 100 lines of 132 characters per line. The adapter also provides for the control, timing and status monitoring of the printer.

User visibility in addressing printer functions is restricted to ESC control code sequences which activate the printer.

ATTRIBUTES

The following control code sequences in mark form indicate the specific attribute to be stored at the current cursor position.

><ESC s	"visual", screen display in normal intensity
><ESC s{A|a}	"forms", restricts entry to alpha-characters and normal punctuation marks for editing
><ESC s{B|b}	"visual", data and underscore to blink, inverse video is not affected
><ESC s{D|d}	"forms", restricts entry to decimal digits and "space" negates ><ESC s{N|n}
><ESC s{E|e}	"forms", entry required before tabbing from the field

VIP7804
continued

ATTRIBUTES (continued)

><ESC s{F|f} "forms", field must be completely filled before tabbing

><ESC s{H|h} "visual", data entered is not to be displayed, inverse video and underscore are not affected

><ESC s{I|i} "visual", inverse video

><ESC s{J|j} "forms", field entry to be right justified

><ESC s{L|l} "visual", screen display in low intensity

><ESC s{M|m} "forms", field to be transmitted only if its contents is changed negates ><ESC s{T|t} , sets ><ESC s{U|u}

><ESC s{N|n} "forms", restricts field to signed decimals and value indicators negates ><ESC s{D|d}

><ESC s{O|o} "printer", suppresses printing, also see "Printer"

><ESC s{P|p} "forms", defines protected field without changing other attributes modifies ><ESC s{M|m} and ><ESC s{U|u}

><ESC s{R|r} "restore", negates attributes set for a field without altering the field assignment

><ESC s{T|t} "forms", transmits the contents of a protected field

><ESC s{U|u} "forms", defines unprotected field whereby data is to be systematically transmitted and made accessible to the operator

><ESC s _ "forms", display underscore

To set up a form, proceed as follows,

- Clear the screen by sending the control code sequence ><ESC `
- Set the terminal in forms mode by the sequence ><ESC [h
- Position the cursor at the start of the 1st field, see "Cursor"
- Enter the "forms" attribute and then the contents of the field
- Position the cursor at the start of the next field
- Repeat the operation of entering the "forms" attribute, then the contents
- Continue the same procedure until the form is completed.

To delete any attribute, send the control code sequence ><ESC [Q

CURSOR : also see "Tabulation"

><BS▽ moves cursor 1 character position backwards on same line, regardless of fields transversed; cursor does not move out of position 1

><CR▽ reset cursor at left margin, character position 1, on same line

><ESC A moves cursor upwards by 1 line, same character position; when in line 1, cursor moves to line 24

><ESC B moves cursor downwards by 1 line, same character position; when in line 24, cursor moves to line 1

><ESC C moves cursor forwards by 1 character position along same line; when out of position 80 of line 24, cursor moves to "home" position

><ESC D moves cursor backwards by 1 character position along same line; when out of "home" position, cursor moves to position 80 of line 24

><ESC f positions cursor according to character position and line number

><ESC H moves cursor to "home" position

><ESC n terminal to indicate its cursor position in the status line, according to the format given in ><ESC f

><LF▽ moves cursor downwards by 1 line; if cursor is aleady in line 24,
- in "roll" mode, data is rolled up by 1 line
- in "non-roll" mode, cursor does not move and "data-overflow" occurs

To position the cursor, proceed as follows using the ASCII table

- Determine the value to be given for the character position, say, 14
  - Start from ASCII code 20 which is a "space"
  - Count 14 codes from ASCII code 20 inclusive
  - The ASCII code arrived at is 2D
  - The EBCDIC equivalent is 60 or graphic -
- Determine the value to be given to the line number, say, 20
  - Use exactly the same method as above for character position
  - The EBCDIC equivalent is F3 or graphic 3
- In the MCS application, send a control sequence in one of the formats :

><ESC f - 3 using graphic symbols

><ESCf><C60><CF3 using EBCDIC values in control character form

EDITING

><ESC J erases to end of page starting from current cursor position, tabs and attributes remaining unaffected; in forms mode, only unprotected data is erased

><ESC K erases to end of field starting from current cursor position; not valid if cursor is already in character position 80

><ESC [I allows characters to be inserted at current cursor position; data surpassing line width is truncated

><ESC [J negates ><ESC [I

><ESC [L inserts single blank line at line indicated by cursor; invalid in forms mode or when cursor is in the status line

><ESC [M deletes line in which cursor is located, including all attributes

><ESC [P deletes character within a line whereby all text is closed up to the left, tabs and attributes remaining unaffected

KEYBOARD

><ESC [X locks keyboard, disables all keyboard functions except for RIS, RES, AUTO-LF, LOCAL and BREAK

><ESC [W unlocks keyboard, negates ><ESC [X

TABULATION : also see "Cursor"

><ESC p sets tab stop at current cursor position; invalid in forms mode

><ESC [g clears tab stop, negates ><ESC p ; invalid in forms mode

><ESC [N clears all tabs irrespective of cursor position; invalid in forms mode

><ESC [Z moves cursor backwards to the previously defined tab position

><HT▽ moves cursor forwards to next character position at which the tab was set; if in forms mode, cursor moves to start of next unprotected field

MODE

><ESC F resets the line graphics mode, negates ><ESC G

><ESC G sets terminal in line graphics mode, in which any of the 11 line graphic symbols can be used

><ESC I causes terminal to transmit next block when in block transmission mode

><ESC q resets the roll mode, negates ><ESC r

><ESC r sets terminal in roll mode unless terminal is in forms mode

><ESC [E resets the block transmission mode

><ESC [block1 block2 block3 block4 F
sets terminal in block transmission mode according to the block-size designated

><ESC [j sets terminal in forms mode, whereby data can only be entered in unprotected fields

><ESC [l sets terminal in text mode, whereby data can be entered anywhere

><ESC [S transmits successive blocks of data automatically when in block transmission mode after the previous block has been ACKed

><ESC [T allows transmission of the block only on receipt of ><ESC I

><ESC ` resets screen to initial state as follows,
- erase the screen
- set cursor to "home" position
- set text mode
- clear all attributes
- set normal intensity for screen illumination

VIP7804
continued

PRINTER

><ESC [0 p — causes data space to be printed in both transmission modes; areas to be suppressed are selected by ><ESC s {0|o} "printer" attribute;
control codes and timings are provided by the printer adapter

><ESC [2 p — terminates printing

><ESC [3 p — prints data stream without affecting screen or data space; control codes ><FF∇, ><VT∇, ><LF∇ and ><CR∇ are provided by the user, with the printer adapter inserting the appropriate "time-fills" for the corresponding codes;
><BS∇ and ><HT∇ must not be sent

><ESC [4 p — prints transparent data without affecting screen or data space; used for control codes other than ><FF∇, ><VT∇, ><LF∇ and ><CR∇ ;
"time-fills" for control codes are provided in the form of ><ESC [? nnn p , where nnn is a 3-digit decimal value

><ESC [5 {0|1} p
sets print mode, as follows,
- if 0, only unprotected fields are printed
- if 1, both protected and unprotected fields are printed

><ESC [7 n p — specifies the number of copies to be printed, where n is a 1-digit decimal value

><ESC [< p — indicates the end of data stream

><ESC [= c p — sets the control code to be used by the printer, where c is the graphic symbol, from 0 through ?, representing the following control code combinations,

0	start CR	end CR		8	start CR	end CR-FF	
1	CR-LF	CR		9	CR-LF	CR-FF	
2	CR-FF	CR		:	CR-FF	CR-FF	
3	CR-VT	CR		;	CR-VT	CR-FF	
4	CR	CR-LF		<	CR	CR-VT	
5	CR-LF	CR-LF		=	CR-LF	CR-VT	
6	CR-FF	CR-LF		>	CR-FF	CR-VT	
7	CR-VT	CR-LF		?	CR-VT	CR-VT	

SECTION VII

QUEUE MAINTENANCE

The queue maintenance utility is described in terms of

- Input Data
- Output Data
- Commands
- Executing QMAINT

INPUT DATA

Input data to QMAINT is in the form of QMAINT commands which are introduced

- either on cards forming an input enclosure in a deck of JCL statements
- or as a subfile retrieved from a source library.

If the QMAINT commands are used repeatedly for such functions as displaying or purging the contents of the queues systematically at the end of a communications session, they should be stored as a member of a library.

OUTPUT DATA

Output data from QMAINT is in the form of print-out reports which are

- SYSOUT report which provides the following
  - lists the QMAINT commands
  - lists the actions resulting from each command in the order listed in the run-time JCL
  - indicates any errors detected during the execution of the QMAINT step
- JOR, job occurrence report, which contains messages defining
  - system errors for which the job has aborted
  - user errors for which the job has halted abnormally.

COMMANDS

QMAINT commands are dealt with in terms of

- Symbolic Convention
- Command Description

Symbolic Convention

The following rules define the QMAINT symbolic convention,

- Keywords for command names and parameters are written in capitals in the text.
- The following symbols must not be specified in user-defined values,

,	comma	/	slash	(	open parenthesis
∇	space	=	equals	)	close parenthesis
;	semi-colon	*	asterisk	"	double quotation

- Utility-reserved keywords must not be specified as user-defined terms.
- A command can span more than 1 card, for as long as the last card containing the end of the command is terminated by a semi-colon.
- Individual parameters of a command may be separated by commas, spaces, or commas and spaces.
- Blank cards and "comment" cards are not processed.
- A user-supplied value exceeding the range of permitted values is disallowed and flagged as an error.

Command Description

Each command is described in terms of

- definition, giving the purpose and function of the command
- format of command, indicating mandatory, positional and optional parameters
- description of parameters, describing the use and restrictions of each parameter
- command report, which lists all the actions performed by QMAINT as a result of the command execution.

QMAINT
Command Description

Command	Definition	Page
COMM	defines a "comment" and may appear anywhere in the sequence of commands.	7-04
PRINT	prints out, without altering, the contents of one, several or all of the queues defined within the network.	7-05
PURGE	destroys all or part of the messages that are completely queued in a given queue.	7-07
QSTATUS	lists the current status and the generation parameters of one, several or all of the queues within the network.	7-08
SEND	sends user-defined messages to the queue.	7-11
STATUS	continues or suspends the processing of QMAINT commands when an error has previously occurred.	7-14

COMM

Definition

The COMM command defines a comment and may appear anywhere in the sequence of commands.

Format of Command

```
COMM  "string"  ;
```

Description of Parameter

string
- a character string enclosed within double quotation marks that must be opened and closed on the same card.

  The string cannot be spawned on more than 1 card, i.e., the double quotation mark closing the string must be on the same card as the double quotation mark opening the string.

  A maximum of 72 characters can be specified for each COMM command.

  If a comment is longer than 72 characters, the excess number of characters must appear on the next COMM command.

Command Report

Only on command listing.

PRINT

Definition

The PRINT command is used to print out, without altering, the contents of one, several or all of the queues defined within the network.

The contents of a queue is the set of messages that are completely queued, that is, not in the transitional state of being sent or received.

The messages are printed out on the order that they would be received from the queue concerned by an application or by BTNS.

Format of Command

```
PRINT { queue-name-1 [ , queue-name-2  ...  queue-name-n ] } ;
      { *                                                 }
```

Description of Parameters

queue-name
- ranges from 1 through 12 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Network Generation Manual.

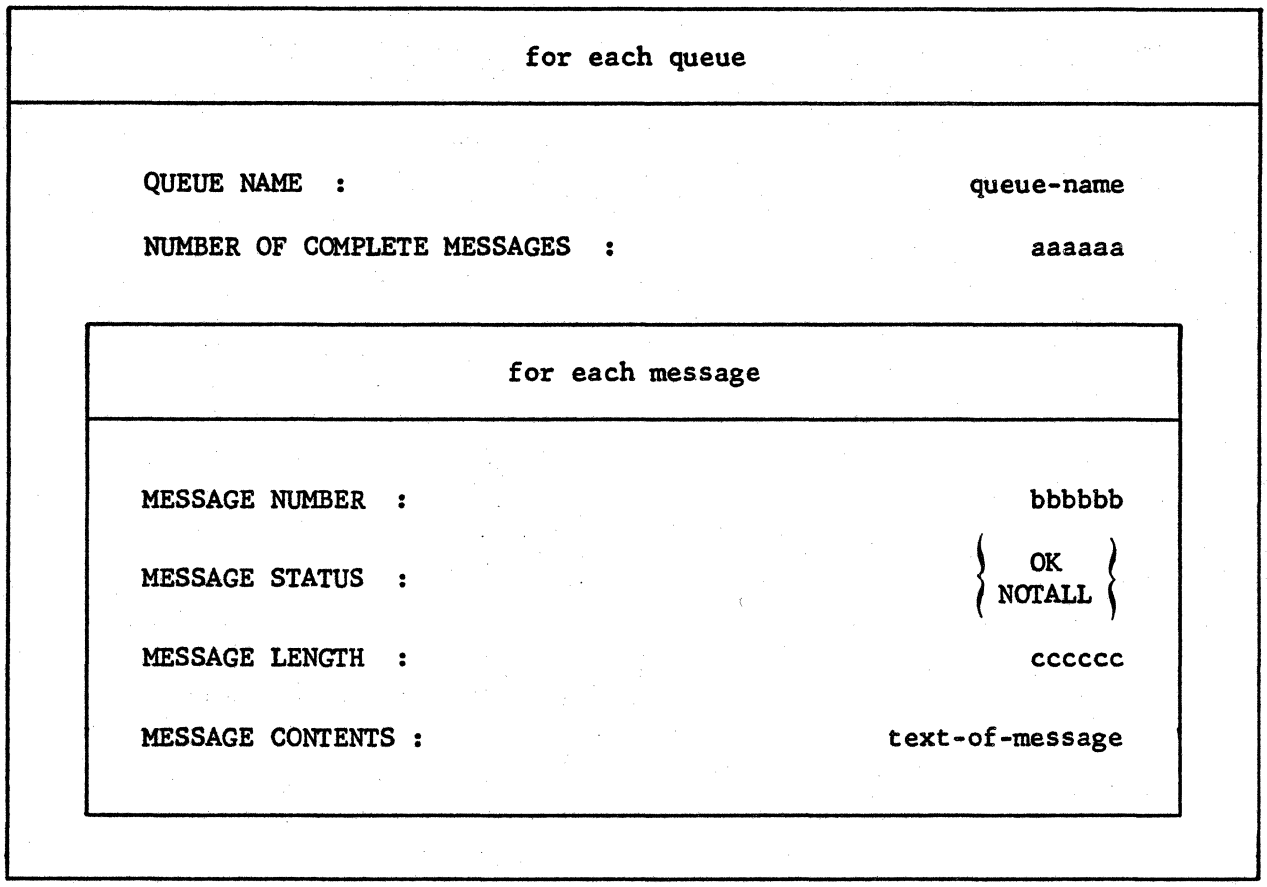
queue-name-1...queue-name-n
- defines the list of queues and the order in which messages are to be printed out.

*
- requests the print-out of the contents of all the queues defined within the given network.

  The order in which the messages are to be printed out will be the order in which the queues were declared at network generation.

PRINT
continued

Command Report



queue-name
- ranges from 1 through 12 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command.

aaaaaa
- number of complete messages in the queue that have been printed.

bbbbbb
- number of the message ranking in the queue.

cccccc
- length of the message in characters.

OK
- complete text for the message has been printed.

NOTALL
- partial text for the message has been printed.

PURGE

Definition

The PURGE command destroys all or part of the messages that are completely queued in the referenced queue.

Format of Command

```
PURGE  queue-name  { ALL              } ;
                   { NUMBMSG = nnnnn  }
```

Description of Parameters

queue-name
- ranges from 1 through 12 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Network Generation Manual.

ALL
- specifies that all the messages present in the queue are to be destroyed.

NUMBMSG
- specifies the number of messages in the queue to be destroyed.

  The number of messages ranges from 1 through 99999.

Command Report

```
QUEUE NAME  :                                  queue-name

NUMBER OF DELETED MESSAGES  :                    aaaaaa
```

queue-name
- ranges from 1 through 12 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command.

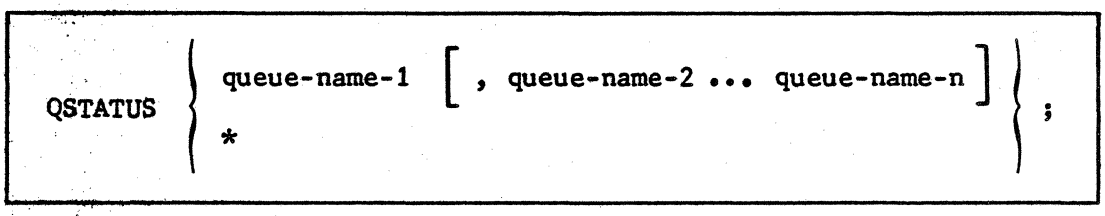
aaaaaa
- number of messages destroyed, see ALL and NUMBMSG above.

QSTATUS

Definition

The QSTATUS command lists the current status and the generation parameters of one, several or all of the queues within the network.

Format of Command



Description of Parameters

queue-name
- ranges from 1 through 12 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Network Generation Manual.

queue-name-1...queue-name-n
- defines the list of queues and the order in which their status and generation parameters are to be listed.

*
- requests the listing of the current status and generation parameters of all the queues defined within the network.

  The order in which this information is listed will be the order in which the queues were declared at network generation.

Command Report

for each queue

```
QUEUE NAME  :                                          queue-name

NUMBER OF COMPLETE MESSAGES  :                           aaaaaa

NUMBER OF MESSAGES IN SEND PHASE  :                      bbbbbb

NUMBER OF MESSAGES IN RECEIVE PHASE  :                   cccccc

[ {NUMBER OF BLOCKS ALLOCATED TO THIS QUEUE  :           dddddd }
  {NUMBER OF BLOCKS USED FOR THIS QUEUE  :               eeeeee }

  {MAXIMUM NUMBER OF BLOCKS IN THE POOL  :               ffffff }
  {NUMBER OF BLOCKS USED FROM THE POOL  :                gggggg } ]

{PROGRAM QUEUE  }
{TERMINAL QUEUE }

QUEUE ATTRIBUTES  :                                      {CORE}
                                                         {DISK}
                                                         [/BREAK]
                                                         [/RESTART]
                                                         [/TWA]
```

queue-name
- ranges from 1 through 12 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command.

aaaaaa
- number of messages completely queued in the current state.

bbbbbb
- number of messages partially sent to the queue, that is, messages not terminated by either EMI or EGI.

QSTATUS
continued

Command Report (continued)

ccccccc
- number of messages partially received from the queue.

dddddd
- number of memory or disk blocks allocated to the queue at network generation through the respective parameters of the corresponding QUEUE command,
  - NUMBLK : number of memory blocks to be used as the memory queue pool
  - NUMREC : number of blocks to be used as the disk queue file.

  NUMBER OF BLOCKS ALLOCATED TO THIS QUEUE appears for memory queues not defined with QCPOOL and for disk queues defined with NUMREC.

eeeeee
- number of used blocks among the "dddddd" blocks reserved, see above.

  NUMBER OF BLOCKS USED FOR THIS QUEUE appears for memory queues not defined with QCPOOL and for disk queues defined with NUMREC.

ffffff
- total number of memory blocks of the memory queue pool to be shared by all queues qualified by the QCPOOL option in their respective QUEUE commands. The total number of memory blocks is defined by the QCPOOL parameter of the GENCOM command.

  MAXIMUM NUMBER OF BLOCKS IN THE POOL appears for memory queues defined with QCPOOL and for disk queues not defined with NUMREC.

gggggg
- number of used memory blocks from the "ffffff" blocks reserved, see above.

  NUMBER OF BLOCKS USED FROM THE POOL appears for memory queues defined with QCPOOL and for disk queues not defined with NUMREC.

BREAK
- applicable only to program-queues, see QUEUE command of Network Generation Manual.

CORE
- if NUMBLK or QCPOOL are specified in the QUEUE command.

DISK
- if NUMBLK and QCPOOL do not appear in the QUEUE command.

RESTART
- applicable only to disk-queues, that is, program-queues and terminal-queues, see QUEUE command of Network Generation Manual.

TWA
- applicable only to program-queues, see QUEUE command of Network Generation Manual.

SEND

Definition

The SEND command sends user-defined messages to the queue.

The text of the message to be sent immediately follows the command and can appear on several cards, each card spanning from column 1 through column 80.

This function is used to simulate terminals and for debugging MAM applications.

Format of Command

```
SEND  queue-name [ , ENDMSG = "end-of-message-string" ][ , LENGTH = nnnn ] ;
```

Description of Parameters

queue-name
- ranges from 1 through 12 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Network Generation Manual.

ENDMSG
- ranges from 1 through 5 alphanumeric characters enclosed within double quotation marks and denotes the "marker" to be used to specify the end of the text of the message to be sent to the queue.

  If ENDMSG is omitted, the message text must be terminated by a //EOM card.

  Either ENDMSG or //EOM must be used.

LENGTH
- defines the length of the message in characters to be sent to the queue. If the length specified conflicts with the number of data cards after the command, a warning ERROR QC 0306 is displayed by QMAINT, see Appendix C.

SEND
continued

Command Report

QUEUE NAME :	queue-name
MESSAGE STATUS :	{ OK / NOTALL }
MESSAGE LENGTH :	aaaaaa
MESSAGE CONTENTS :	text-of-message

queue-name
- ranges from 1 through 12 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command.

aaaaaa
- number of characters in the message sent to the queue.

OK
- the number of characters in the "text-of-message" matches the number specified by "aaaaaa".

NOTALL
- the number of characters in the "text-of-message" is less than the number specified by " aaaaaa".

text-of-message
- the text of the message sent to the queue is edited in a maximum of 110 characters per line.

Example 1
LENGTH not specified

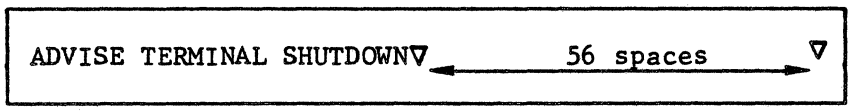
1. a

```
END
ADVISE TERMINAL SHUTDOWN
SEND Q1,ENDMSG="END";
```

1. b

```
//EOM
ADVISE TERMINAL SHUTDOWN
SEND Q1;
```

In 1. a and 1. b, the message to be sent to queue Q1 will be 80 characters, despite the fact that the actual message contains only 24 characters.



This is because the LENGTH parameter in the SEND command was not specified and therefore the entire 80 columns of the card containing the message text after the SEND command is sent to the queue.

Example 2
LENGTH specified

2. a

```
END
ADVISE TERMINAL SHUTDOWN
SEND Q1,ENDMSG="END",LENGTH=24;
```

2. b

```
//EOM
ADVISE TERMINAL SHUTDOWN
SEND Q1,LENGTH=24;
```

In 2. a and 2. b, the message to be sent to queue Q1 will be 24 characters.

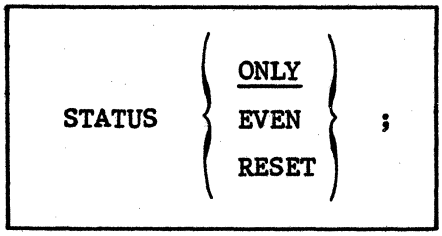
Using this method of specifying the LENGTH parameter, the message can be "tailored" exactly and no space is therefore wasted in the queue.

STATUS

Definition

The STATUS command continues or suspends the processing of QMAINT commands when an error has previously occurred.

Format of Command



Description of Parameters

ONLY
- this is the default, whereby the commands are only executed provided that no error has occurred.

EVEN
- only the following command is executed when an error has occurred.

RESET
- resets the error count to zero.

Command Report

Only on command listing.

EXECUTING QMAINT

Executing QMAINT is dealt with in terms of,

- Run-time Prerequisites
- Run-time JCL

Run-time Prerequisites

All the following prerequisites must be met to execute QMAINT, namely,

- A previous CNC session describing all the queues referenced by the QMAINT commands must have been successfully run.
- Each program-queue referenced by a QMAINT command must be available, namely,
  - it must not be allocated to any application in IN, INOUT or OUT modes
  - it must not have any terminals connected to it
- Each terminal-queue referenced by a QMAINT command must be available, that is, the queue must not be currently allocated to any application
  - either explicitly through a $QASSIGN statement
  - or implicitly through the terminal connection to the application.

Run-time JCL

The syntax for QMAINT run-time JCL is as follows,

- STEP statement
  - H_QMAINT is the system load-module in the system load-module library called SYS.HLMLIB and must be specified as shown.
- ASSIGN statement
  - H_CR is the system-reserved internal-file-name for the file containing the QMAINT commands, either as an input enclosure or as a source library member, and must be specified as shown.

In the "Example of QMAINT Execution",

- the job QDISP performs actions on the queues specified within the current network
- the maintenance actions to be performed are described by the QMAINT commands in the input enclosure.

For detailed explanation, see Appendix B.

QMAINT
Run-time JCL

commands retrieved
from an input enclosure

```
$JOB job-name , USER=user-name , PROJECT=project-name ;

STEP H_QMAINT , SYS.HLMLIB ;

ASSIGN H_CR , *input-enclosure-name ;

ENDSTEP ;

$INPUT input-enclosure-name [ , TYPE=DATASSF ] ;

                            { QMAINT  }
                            { commands }

$ENDINPUT ;

$ENDJOB ;
```

commands retrieved
from a source library member

```
$JOB job-name , USER=user-name , PROJECT=project-name ;

STEP H_QMAINT , SYS.HLMLIB ;

ASSIGN H_CR , external-file-name , SUBFILE=member-name ,
       DEVCLASS=device-class-name , MEDIA=media-name ;

ENDSTEP ;

$ENDJOB ;
```

QMAINT Run-time JCL
(continued)

Example
of
QMAINT Execution

```
$JOB QDISP , USER = UNAME , PROJECT = WAGE ;

STEP H_QMAINT , SYS. HLMLIB ;

ASSIGN H_CR , *QINP ;

ENDSTEP ;

$INPUT QINP ;

    COMM "SPECIFIED QUEUES ARE Q1,Q2,Q3,Q4" ;
    COMM "DISPLAY STATUS OF ALL THE QUEUES" ;
    QSTATUS * ;
    COMM "PURGE Q1 (ALL MESSAGES) AND Q2 (ONLY 5 MESSAGES)" ;
    PURGE Q1 , ALL ;
    PURGE Q2 , NUMBMSG = 5 ;
    COMM "CONTINUE EVEN IF WRONG RESULT FROM PURGE" ;
    STATUS EVEN ;
    COMM "PRINT CONTENTS OF QUEUE Q3" ;
    PRINT Q3 ;
    COMM "SEND ONE 17 CHARACTER MESSAGE TO Q4" ;
    SEND Q4 , ENDMSG = "ENDMS" , LENGTH = 17 ;
    TERMINAL TO LOGON
    ENDMS

$ENDINPUT ;

$ENDJOB ;
```

SECTION VIII

DYNAMICS OF COMMUNICATIONS

Events occurring during a communications session are governed both by the user and the system.

The determining factors are,

- Execution chronology of the software components
- Levels of simultaneity for communications
- Optimum priorities for the software components
- Data flow during message exchange
- Allocating memory resources.

EXECUTION CHRONOLOGY OF THE SOFTWARE COMPONENTS

Before any communications session can take place, the network environment must first be created, using the CNC utility.

The network is successfully created if no errors occur during generation.

BTNS and FNPS can be started to allow VCAM subsystems to execute. QMON can then be started to allow, in turn, MCS applications to execute. QMON runs as a separate service job from both BTNS and FNPS.

Whenever backing store is destroyed, the following actions must be performed,

- the CNC utility must be rerun
- the option MAM=YES or MAM=REFORMAT must be specified at system initialization if disk queueing is involved (MAM=YES is the default).

Backing store is destroyed for one of following reasons,

- either through a disk failure
- or when the CLEAN option is specified at restart.

Further constraints in determining the order in which the software components are executed, are

- the CNC utility cannot be run when any communications component is currently executing, and vice versa
- BTNS cannot be run when another occurrence of BTNS is currently executing
- an MCS application step with a $QASSIGN statement specifying a queue currently allocated to another step cannot be run.

Failure to comply with any of the contraints listed on the previous page, will lead to a step abort.

Up to 4 occurrences of the FNPS service can be started to execute simultaneously.

Each occurrence is identified by its associated "fnp-name" declared in the FNP command of the CNC generation. A maximum of 4 FNP commands can be so declared.

LEVELS OF SIMULTANEITY FOR COMMUNICATIONS

Within the limits of operability as previously defined, communications components can be started and terminated for as long as the maximum system multiprogramming level is not exceeded.

The following occurrences illustrate the levels of simultaneity for a communications session over the BTNS/URP secondary network,

1. A data collection MCS application DATCOLL starts.
   Its function is to empty disk input queues filled by BTNS during a previous session.

   Number of simultaneities : 1

2. A data distribution MCS application DATDIST starts.
   Its function is to distribute to output queues, messages generated by a batch program during a previous session.

   Number of simultaneities : 2

3. A TDS job is started.
   Connections from the network are not yet possible, although TDS is available to batch entries.

   Number of simultaneities : 3

4. A TDS batch entry starts.
   The batch entry requests connection to TDS to execute file updates.

   Number of simultaneities : 4

5. DATDIST has completed distributing all messages to output queues and terminates.

   Number of simultaneities : 3

6. A file enquiry MCS application FILEINQ starts.
   The application awaits requests to be received into its input queues.

   Number of simultaneities : 4

7. BTNS now starts, which results in the following,

   - the BTNS/URP secondary network is initialized through the "ST gencomname" system console command
   - QMON is then activated through the "ST QMON" network control command. ★
   - log-on requests from terminals to connect to defined input queues and TDS are accepted
   - the distribution of data enqueued by DATDIST to the terminals connected to input queues is now started.

   Number of simultaneities : 4

   Although BTNS and QMON run as separate service jobs, they do not occupy any level of simultaneity.

8. DATCOLL has completed emptying the disk queues and terminates.

   Number of simultaneities : 3

9. The TDS batch entry, see step 4, has updated its files and terminates.

   Number of simultaneities : 2

10. TDS now terminates as it is no longer required.

    Number of simultaneities : 1

11. FILEINQ has accepted all enquiries from the input queues and terminates.

    Number of simultaneities : 0

12. BTNS is retained in the system to fill the disk queues.
    This operation would be continued in a following session by starting up the application DATCOLL, see step 1.

    Number of simultaneities : 0

13. A shutdown is issued.

    BTNS terminates and the communications session ends.

OPTIMUM PRIORITIES FOR SOFTWARE COMPONENTS

In order to maintain efficient response times, appropriate dispatching priorities for the various communications components must be selected.

Batch jobs are given low priorities as they are not subject to real-time constraints.

The user specifies a job class which determines the following for the job:

- its scheduling priority
- its dispatching priority
- its associated level of multiprogramming, being the number of jobs executable at one time.

An example of such considerations is the following situation:

- CNC and QMAINT utilities, being normal batch jobs, are not subject to any response times and can therefore be executed as jobs of class, say, P
- However, an MCS application step and the TDS service, when executed concurrently, that is, both being available in the system at the same time, should have the same dispatching priority.

  While the programmer has no control over VCAM queues for the TDS service, the transactions themselves can be written in such a way as to optimize multitasking, see TDS Documentation.

  For the MCS application, in order not to degrade system performance, unnecessary scanning of the program-queue should be avoided, examples of such indiscrimnate scanning are
  - a RECEIVE (H_RECEIVE) with "no data"
  - an ACCEPT (H_MSGCNT) instead of building a suitable queue structure and using one of the scanning techniques recommended.

  By specifying J as the job class for the TDS service and H for that of the MCS application, the scheduling and execution priorities are 6 and 0, respectively.

  These values are the recommended defaults for the DPS 7 installation.
- in this case, the level of multiprogramming for the job classes specified for the TDS service and the MCS application is 1, however, this value can be modified by the MS system console command, see System Operator's Guide.

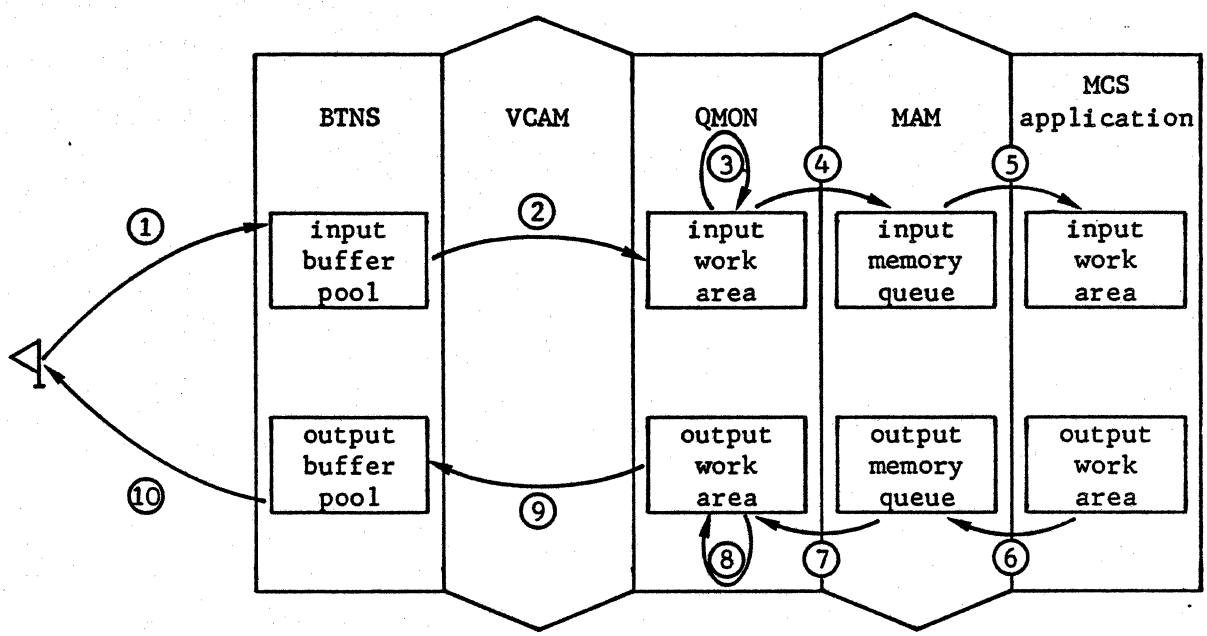
For a description of the use of job class, see System Administrator's Manual.

DATA FLOW DURING MESSAGE EXCHANGE

Message exchange follows a prescribed path and involves the following types,

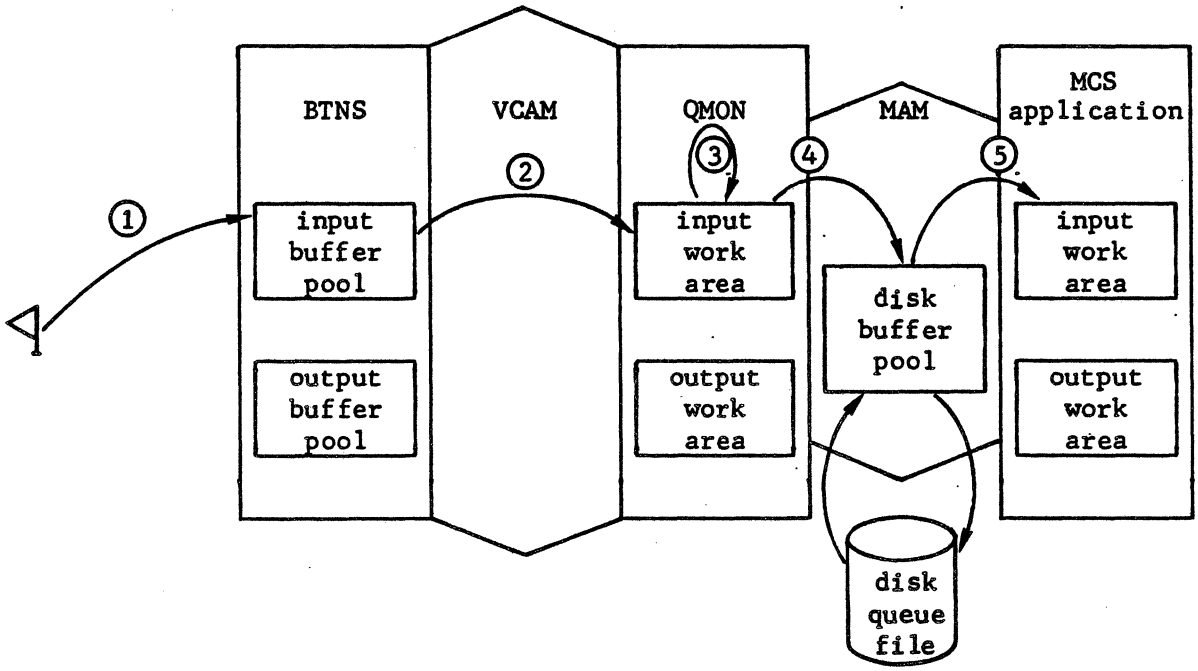
- exchange between an MCS application and a terminal using a memory queue
- exchange between an MCS application and a terminal using a disk queue
- exchange between a VCAM subsystem (communications service) and a terminal.

Example of Exchange
Between an MCS Application and a Terminal
Using a Memory Queue



1. When the line is polled, the terminal sends a message which is stored in the BTNS buffer pool.
   BTNS dynamically allocates buffer units to contain the whole message.
2. The message is sent by BTNS to QMON via VCAM.
3. Control characters and marks are translated by QMON according to terminal type and data format for input.
4. QMON stores the message in the memory input queue to which the terminal is connected by issuing a call to the MCS "send" procedure.
5. When the application issues a "receive" to the symbolic input queue, the message or a part of it, is moved into the application's input work area. As many "receive's" as necessary are issued to collect the whole message.
6. When the message is to be sent to the terminal, the application issues a "send" to the symbolic output queue.
   The message is transferred from the application's output work area to the memory output queue associated with the terminal.
7. When the output request is located, QMON issues a call to the MCS "receive" procedure to store the message in the QMON area.
8. Control characters and marks are translated by QMON according to terminal type and data format for output.
9. The message is sent by QMON to BTNS via VCAM.
10. The message is transferred from the BTNS buffer pool to the terminal.

Example of Exchange
Between an MCS Application and a Terminal
Using a Disk Queue



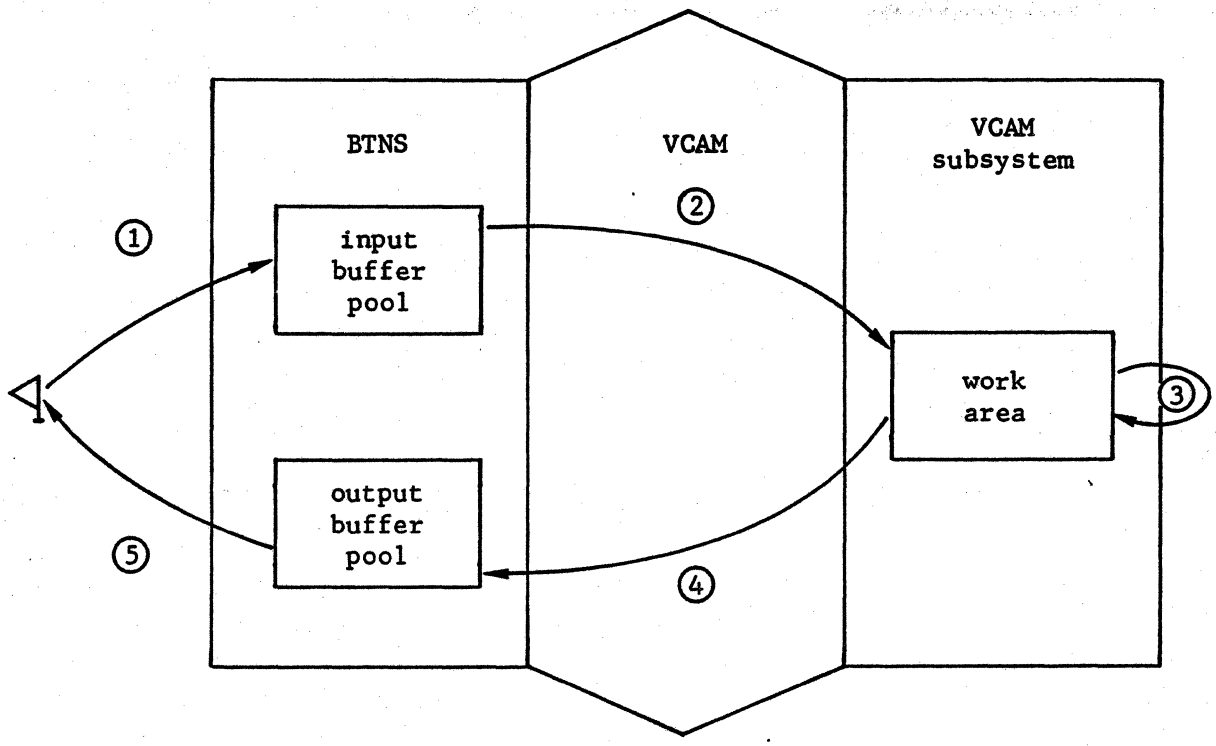
The example shows input from the terminal to the MCS application.
The procedure is identical in reverse from the application to the terminal.

1. When the line is polled, the terminal sends a message which is stored in the BTNS buffer pool.
   BTNS dynamically allocates buffer units to contain the whole message.
2. The message is sent by BTNS to QMON via VCAM.
3. Control characters and marks are translated by QMON according to terminal type and data format for input.
4. When QMON issues a call to the MCS "send" procedure to store the message in the disk input queue to which the terminal is connected, the following events take place,
   - the message is first moved to the disk I/O buffer pool
   - the message is then written to the disk queue file.

   Disk access for QMON is asynchronous which means that QMON does not have to wait for the completion of an I/O operation.
5. When the application issues a "receive" to the symbolic input queue, the following events occur,
   - the message is read from the disk queue file into the disk I/O buffer pool
   - the message is then moved into the application's input work area

   Disk access for the application is synchronous which means that the application must wait for the completion of an I/O operation.

Example of Exchange
Between a VCAM Subsystem and a Terminal



GCOS Communications Services, collectively termed VCAM subsystems, are,

- RBF6/FTF6, Remote Batch Facility & File Transfer Facility, Mini 6-DPS 7
- DJP/DFT, Distributed Job Processing and DSA File Transfer, DPS 7-to-DPS 7
- IOF, Interactive Operator Facility
- TDS, Transaction Driven Subsystem
- CARDLESS, known to the system as READER
- TILS, Transactional and Interactive Load Simulator
- OLTD, On-Line Tests and Diagnostics

1. A message is sent by the terminal to be stored in the BTNS buffer pool, occupying as many buffer units as required.
2. The message is moved to the work area designated by the VCAM subsystem.
3. The message is handled by the VCAM subsystem.
   In the case of TDS, for example, the message activates or is processed by a transaction program or a set of transaction processing routines.
4. A reply to the terminal is moved from the VCAM subsystem work area to the output buffer pool.
5. The message is transferred from the BTNS buffer pool to the terminal.

ALLOCATING MEMORY RESOURCES

Multiprogramming in a virtual memory environment may lead to an overload situation where several applications compete for memory resources.

The effects on the communications session are

- that the segments of its components not currently executing may be swapped with batch programs
- that these segments, when needed to process a message on arrival, must be reloaded, causing
  - a tremendous increase in system overhead
  - a considerable increase in response time
  - a degradation of overall throughput.

The problems to be solved are

- avoiding memory overload of the system:

  The sum of the "working-sets" of applications executing concurrently must not exceed the physical memory size available.

- guaranteeing the minimum memory resources:

  In order to ensure rapid "turn-around", segments of communications components needed to process specific data, must be retained in memory even if inactive over long periods.

The solutions are

- using the SIZE statement:

  The SIZE statement declares the "working-set" of the application for the purpose of controlled scheduling to avoid memory overload.

- using the MAXMEM and MINMEM parameters of the STEP statement, as follows,

  MAXMEM:
  Used for tuning the "working-set" and should be discontinued when the optimal size has been determined.
  This facility ensures that
  - the amount of memory allocated will never be less than the DWS specified in the SIZE statement
  - the system will not attempt to execute the step if the amount of physical memory available is not greater than or equal to the DWS, even if the step could be run on less.
    The step does <u>not</u> benefit by the gradual release of memory resources as the system load decreases.

  MINMEM:
  Used after the optimal DWS has been determined to allocate permanently to the step a memory resource equal to the DWS whatever the load of the system may be.
  This facility is used when "turn-around" times for the communications session are slow, indicating that the components needed for processing are absent in memory. By guaranteeing "minimum-memory" requirements, all communications components needed for the session will be present in memory until termination.

- using the PMM console command:

  In order to ensure the presence of system functions, the PMM command can be used to "lock" these segments in memory so that these become permanently available.

Allocating memory resources is dealt with in regard to

- MCS applications
- MAM and VCAM
- BTNS (which includes TNS), FNPS and QMON.

Allocating Memory to MCS Applications

The way to determine how memory resources are to be allocated to MCS applications is as follows,

- establishing the size of the DWS
- guaranteeing memory by declaring the DWS as the minimum resource required.

ESTABLISHING THE DWS SIZE

The first time that the step is executed, the dws-value specified for the SIZE statement is calculated from the linkage listing of the application and declared as MAXMEM in the STEP statement.

The JOR listing at the end of the job step will indicate the number of missing segments, if any.

The general rule in tailoring the dws-value specified in units of K bytes is

- if few missing segments are indicated, a smaller dws-value can be specified until such a time when an increase in missing segments occurs
- if many missing segments are indicated, a proportionately higher dws-value should be specified, until such time that the first condition is reached.

Successive executions of the job step will ultimately give an optimum dws-value.

GUARANTEEING MEMORY

The "optimum" dws-value is then specified for the SIZE statement and declared as MINMEM in the STEP statement.

By this means, the step is "guaranteed" the minimum memory resource before it is started by the system.

```
run-time JCL for DWS

STEP ... { MAXMEM } ;
         { MINMEM }

SIZE dws-value ... ;
```

Allocating Memory to MAM and VCAM

Both MAM and VCAM are system functions which can be "locked" in memory by the PMM console command, the format and function being as follows,

- PMM VCAM : VCAM segments are "locked"
- PMM MAMM : MAM segments managing memory queues are "locked"
- PMM MAMD : MAM segments managing disk queues are "locked".

After the PMM command specifying the function to be "locked" is issued, the function remains in memory until the CMM command specifying the function is issued making the function eligible for swapping out of memory.

Allocating Memory to BTNS, FNPS and QMON

All three service jobs run automatically under MINMEM when initialized by the ST network control command.

They communicate to the system the memory size needed according to the configuration present.

No user intervention is required to ensure the presence of BTNS, FNPS and QMON.

APPENDIX A

MCS APPLICATION EXAMPLE

This is a test program for the "two-way-alternate" option, in which the application sends the message "ENTER M OR G" to the console operator.

The console operator, on receipt of this message, can then reply

- M, in which case the application, by delimiting transmission with EMI, retains the turn to transmit
- G, in which case the application delimits transmission with EGI, thereby giving the turn to transmit to the operator
- E, in which case the application then terminates.

This application is part of a test package for communications software, and serves as an example of the user interface with MCS.

A detailed explanation of the TWA option used with interactive applications is given on page 2-16.

MCS Application Example
in
MCS COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID. TEST_TWA.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. LEVEL-64.
OBJECT-COMPUTER. LEVEL-64.

DATA DIVISION.

  WORKING-STORAGE SECTION.
    01  ACO1REPLY  PIC X.
    01  MSGEMI     PIC X(90) VALUE ALL "EMI".
    01  MSGEGI     PIC X(90) VALUE ALL "EGI".
    01  BUFIN      PIC X(2000).

  COMMUNICATION SECTION.
    CD  CDIN FOR INPUT QUEUE CDIN-QUEUE-NAME.
    CD  CDOUT FOR OUTPUT.
        01 CDOUT-SPECIF.
          02 CDOUT-DESTINATION-COUNT  PIC 9(4) VALUE 1.
          02 CDOUT-TEXT-LENGTH        PIC 9(4) VALUE 90.
          02 FILLER                   PIC XX.
          02 FILLER                   PIC X.
          02 CDOUT-DESTINATION        PIC X(12) VALUE "INTQOUT".

PROCEDURE DIVISION.

  BEGIN.
    ENABLE INPUT CDIN KEY "PASW"
    ENABLE OUTPUT CDOUT KEY "PASW"
    MOVE "INTQIN" TO CDIN-QUEUE-NAME.

  LOOP.
    DISPLAY "ENTER M OR G" UPON CONSOLE
    ACCEPT ACO1REPLY FROM CONSOLE

    IF ACO1REPLY="M"
       SEND CDOUT FROM MSGEMI WITH EMI.

    IF ACO1REPLY="G"
       MOVE 90 TO CDOUT-TEXT-LENGTH
       SEND CDOUT FROM MSGEGI WITH EGI.

    IF ACO1REPLY="E"
       GO TO TERM

    ELSE GO TO LOOP.

  TERM.
    RECEIVE CDIN MESSAGE INTO BUFIN
    STOP RUN.

  end of test for the two-way-alternate option.
```

MCS Application Example
in
GPL

```
TEST_TWA : PROC ;
/* DECLARATIONS */
    DCL H_BUFIN      CHAR(2000) ;
    DCL A_PO4CDOUT   PTR;
    DCL A_PO4CDIN    PTR;
    DCL A_PO4MESI    PTR;
    DCL A_PO4MESO    PTR;
    DCL MSGEMI       CHAR(90)  STATIC  INIT((30)"EMI") ;
    DCL MSGEGI       CHAR(90)  STATIC  INIT((30)"EGI") ;
    DCL A_CO1REPLY   CHAR(1)  STATIC;
    DCL A_F15MOP     FIXED  BIN(15)  STATIC  INIT(12) ;
    DCL A_C12MSG     CHAR(12)  STATIC  INIT("ENTER M OR G") ;
    DCL A_F15MXL     FIXED  BIN(15)  STATIC  INIT(1) ;

    $H_CD  INPUT , PREFIX = 'CDIN_' , ATTRIB = 'STATIC  INTERNAL') ;
    $H_CD  OUTPUT , PREFIX = 'CDOUT_' , ATTRIB = 'STATIC  INTERNAL' ;
/* PROCESS */
    BEGIN ;
      A_PO4CDOUT = ADDR(CDOUT_OUTPUT_CD) ;
      A_PO4CDIN = ADDR(CDIN_INPUT_CD) ;
      A_PO4MESI = ADDR(H_BUFIN) ;
      CDOUT_DESTINATION_COUNT = 1 ;
      CDOUT_QUEUE_NAME(1) = "INTQOUT" ;
      CDIN_QUEUE_NAME(1) = "INTQIN" ;

      $H_ENABLE  A_PO4CDIN , INPUT ;
      $H_ENABLE  A_PO4CDOUT , OUTPUT ;

    LOOP ;
      $H_SENDOR  REPLY = A_CO1REPLY , MAXLEN = A_F15MXL , MESSAGE = A_C12MSG ,
         LENGTH = A_F15MOP , MAIN ;
      IF  A_CO1REPLY = 'M'  THEN  BEGIN ;
         A_PO4MESO = ADDR(MSGEMI) ;
         CDOUT_TEXT_LENGTH = 90 ;
         $H_SEND  A_PO4CDOUT , INADDR = A_PO4MESO , ENDCHAR = EMI ;
         END ;
      IF  A_CO1REPLY = "G"  THEN  BEGIN ;
         A_PO4MESO = ADDR(MSGEGI) ;
         CDOUT_TEXT_LENGTH = 90 ;
         $H_SEND  A_PO4CDOUT , INADDR = A_PO4MESO , ENDCHAR = EGI ;
         END ;
      IF  A_CO1REPLY = "E"  THEN  GOTO  TERM ;  ELSE  GOTO  LOOP ;

    TERM ;
      $H_RECEIVE  A_PO4CDIN , OUTADDR = A_PO4MESI , LENGTH = 200 ;
      END ;
      END  TEST_TWA ;
    end of test for the two-way-alternate option.
```

APPENDIX B

QMAINT SYSOUT REPORT

The results of running the QMAINT utility are,

- QMAINT error messages, see Appendix C
- the QMAINT sysout report.

The purpose of the report is to enable the user to ascertain that the maintenance functions on the contents of his memory and disk queues are correctly carried out.

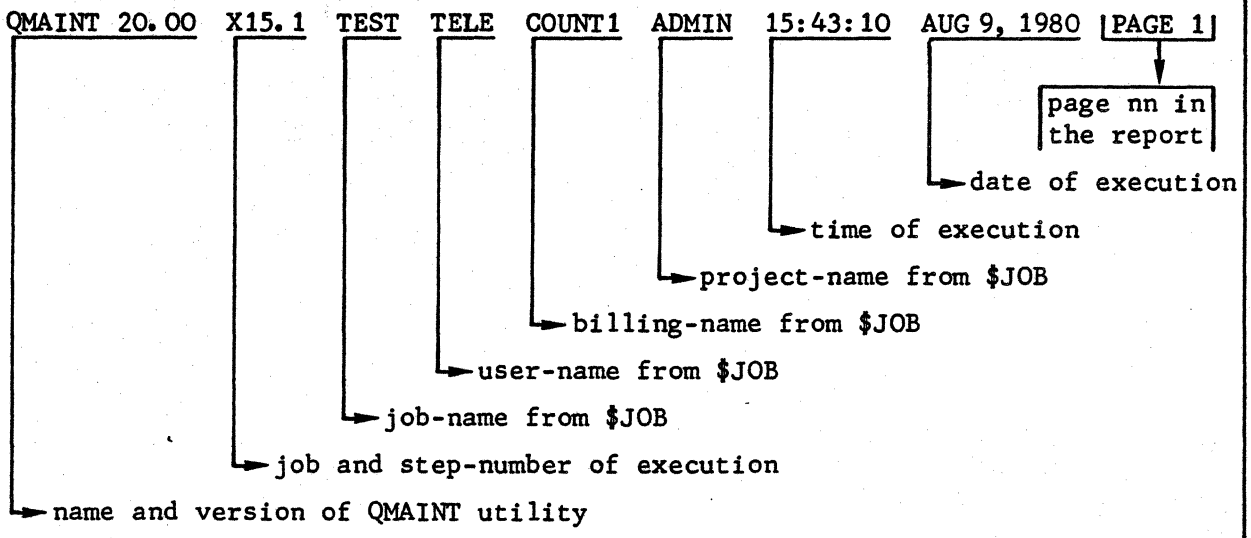
The structure of the QMAINT sysout report is as follows,

- the header line, which appears as the first line of the report and has the standard format for any GCOS utility
- the header banner
- run-time JCL, containing the listing of QMAINT commands provided by the user
- execution report, providing a detailed report of each QMAINT command in the order listed in the run-time JCL, and any error messages as a result of QMAINT execution
- error summary, being a statistical report for each severity.

In the following example, QMAINT executes actions on

- Q1, Q5 and Q6, which are disk queues
- Q2 and Q3, which are memory queues.

Header Line



Header Banner

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
  GCOS L64
                                QMAINT VERSION: 20.00                    AUG 9, 1980
  EXECUTION REPORT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

QMAINT Run-time JCL
Execution Report
see pages following

Error Summary

```
    ERROR QC0314 SEVERITY 00 TOTAL NUMBER OF ERRORS :          0000000000
   *ERROR QC0315 SEVERITY 01 NUMBER OF ERRORS OF SEVERITY 1:   0000000000
  **ERROR QC0316 SEVERITY 02 NUMBER OF ERRORS OF SEVERITY 2:   0000000000
 ***ERROR QC0317 SEVERITY 03 NUMBER OF ERRORS OF SEVERITY 3:   0000000000
```

QMAINT Run-time JCL

```
$JOB TEST , USER=TELE , PROJECT=ADMIN , BILLING=COUNT1 ;
STEP H_QMAINT , SYS.HLMLIB ;
ASSIGN H_CR , *QMAINT ;
ENDSTEP ;
$INPUT QMAINT , TYPE=DATASSF ;
SEND Q1 , ENDMSG="//FIN" , LENGTH=160 ;
AAAAAAAAAAAAAAAAA                                    AAAAAAAAA111
AAAAAAAAAAAAAAAA<----------sample printout---------->AAAAAAAAAAAAAA
AAAAAAAAAAAAAA              of contents              AAAAAAAAAAAAA111
AAAAAAAAAAAA                                        AAAAAAAAAAAAAA111
//FIN
SEND Q1 , LENGTH=160 ;
BBBBBBBBBBBBBBBBB                                      BBBBBBBBBB2
BBBBBBBBBBBBBBBB<----------sample printout---------->BBBBBBBBBBBB2
BBBBBBBBBBBBBB              of contents              BBBBBBBBBBBBBBB
BBBBBBBBBBBB                                        BBBBBBBBBBBBBBBBB
//EOM
QSTATUS *;
PRINT *;
PRINT Q1;
PRINT Q4;
PRINT Q5;
PRINT Q6;
PURGE Q1 , NUMBMSG=1;
PURGE Q1 , NUMBMSG=1;
QSTATUS *;
PURGE Q2 , ALL;
QSTATUS Q1 , Q2;
PRINT Q1 , Q3 , Q5 , Q6;
PURGE Q1 , NUMBMSG=1;
PURGE Q3 , ALL;
PURGE Q4 , NUMBMSG=3;
PRINT Q3;
$ENDINPUT;
$ENDJOB;
```

QMAINT
Execution Report

```
SEND Q1 ,  ENDMSG="//FIN" , LENGTH=160;

           QUEUE NAME  :                                    Q1
           MESSAGE STATUS  :                                OK
           MESSAGE LENGTH  :                                000160
           MESSAGE CONTENTS  :
           AAAAAAAAAAAAAAAA                                   AAAA111
           AAAAAAAAAAAA                                     AAAAAAAAAA
           AAAAAAAAAA          sample printout            AAAAAAAAAA111
           AAAAAAA               of contents           AAAAAAAAAAAAA111

SEND Q1 , LENGTH=160;

           QUEUE NAME  :                                    Q1
           MESSAGE STATUS  :                                OK
           MESSAGE LENGTH  :                                000160
           MESSAGE CONTENTS  :
           BBBBBBBBBBBBBBB                                    BBBBBB2
           BBBBBBBBBBBB                                     BBBBBBBBB2
           BBBBBBBBBB          sample printout            BBBBBBBBBBBB
           BBBBBBB               of contents           BBBBBBBBBBBBBBB
```

```
QSTATUS *;
          QUEUE NAME  :                                   Q1
          NUMBER OF COMPLETE MESSAGES  :                  000002
          NUMBER OF MESSAGES IN SEND PHASE  :             000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :          000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :             032767
          NUMBER OF BLOCKS USED FROM POOL  :              000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                              DISK

          QUEUE NAME  :                                   Q2
          NUMBER OF COMPLETE MESSAGES  :                  000000
          NUMBER OF MESSAGES IN SEND PHASE  :             000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :          000000
          NUMBER OF BLOCKS ALLOCATED TO THIS QUEUE  :     000040
          NUMBER OF BLOCKS USED FOR THIS QUEUE  :         000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                              CORE

          QUEUE NAME  :                                   Q3
          NUMBER OF COMPLETE MESSAGES  :                  000000
          NUMBER OF MESSAGES IN SEND PHASE  :             000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :          000000
          NUMBER OF BLOCKS ALLOCATED TO THIS QUEUE  :     000040
          NUMBER OF BLOCKS USED FOR THIS QUEUE  :         000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                              CORE

          QUEUE NAME  :                                   Q4
          NUMBER OF COMPLETE MESSAGES  :                  000000
          NUMBER OF MESSAGES IN SEND PHASE  :             000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :          000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :             032767
          NUMBER OF BLOCKS USED FROM POOL  :              000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                              DISK

          QUEUE NAME  :                                   Q5
          NUMBER OF COMPLETE MESSAGES  :                  000000
          NUMBER OF MESSAGES IN SEND PHASE  :             000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :          000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :             032767
          NUMBER OF BLOCKS USED FROM POOL  :              000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                              DISK

          QUEUE NAME  :                                   Q6
          NUMBER OF COMPLETE MESSAGES  :                  000000
          NUMBER OF MESSAGES IN SEND PHASE  :             000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :          000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :             032767
          NUMBER OF BLOCKS USED FROM POOL  :              000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                              DISK
```

```
PRINT *;

          QUEUE NAME  :                                        Q1
          NUMBER OF COMPLETE MESSAGES  :                       000002
          MESSAGE NUMBER  :                                    000001
          MESSAGE STATUS  :                                       OK
          MESSAGE LENGTH  :                                    000160
          MESSAGE CONTENTS  :
          AAAAAAAAAAAAAAAA                                        AAAA111
          AAAAAAAAAAAAA <---------- sample printout ----------> AAAAAAAAAA
          AAAAAAAAAA                  of contents              AAAAAAAAAA111
          AAAAAAA                                             AAAAAAAAAAAA111
          MESSAGE NUMBER  :                                       000002
          MESSAGE STATUS  :                                          OK
          MESSAGE LENGTH  :                                       000160
          MESSAGE CONTENTS  :
          BBBBBBBBBBBBBBB                                         BBBBBB2
          BBBBBBBBBBBBB <---------- sample printout ----------> BBBBBBBBB2
          BBBBBBBBBB                  of contents              BBBBBBBBBBBBB
          BBBBBBB                                             BBBBBBBBBBBBBBB

          QUEUE NAME  :                                           Q2
          NUMBER OF COMPLETE MESSAGES  :                          000000

          QUEUE NAME  :                                           Q3
          NUMBER OF COMPLETE MESSAGES  :                          000000

          QUEUE NAME  :                                           Q4
          NUMBER OF COMPLETE MESSAGES  :                          000000

          QUEUE NAME  :                                           Q5
          NUMBER OF COMPLETE MESSAGES  :                          000000

          QUEUE NAME  :                                           Q6
          NUMBER OF COMPLETE MESSAGES  :                          000000

PRINT Q1;

          QUEUE NAME  :                                        Q1
          NUMBER OF COMPLETE MESSAGES  :                       000002
          MESSAGE NUMBER  :                                    000001
          MESSAGE STATUS  :                                       OK
          MESSAGE LENGTH  :                                    000160
          MESSAGE CONTENTS  :
          AAAAAAAAAAAAAAAA                                        AAAA111
          AAAAAAAAAAAAA <---------- sample printout ----------> AAAAAAAAAA
          AAAAAAAAAA                  of contents              AAAAAAAAAA111
          AAAAAAA                                             AAAAAAAAAAAA111
          MESSAGE NUMBER  :                                       000002
          MESSAGE STATUS  :                                          OK
          MESSAGE LENGTH  :                                       000160
          MESSAGE CONTENTS  :
          BBBBBBBBBBBBBBB                                         BBBBBB2
          BBBBBBBBBBBBB <---------- sample printout ----------> BBBBBBBBB2
          BBBBBBBBBB                  of contents              BBBBBBBBBBBBB
          BBBBBBB                                             BBBBBBBBBBBBBBB
```

QMAINT Execution Report
(continued)

```
PRINT Q4 ;

          QUEUE NAME  :                                  Q4
          NUMBER OF COMPLETE MESSAGES  :                 000000

PRINT Q5 ;

          QUEUE NAME  :                                  Q5
          NUMBER OF COMPLETE MESSAGES  :                 000000

PRINT Q6 ;

          QUEUE NAME  :                                  Q6
          NUMBER OF COMPLETE MESSAGES  :                 000000

PURGE Q1 , NUMBMSG = 1 ;

          QUEUE NAME  :                                  Q1
          NUMBER OF DELETED MESSAGES  :                  000001

PURGE Q1 , NUMBMSG = 1 ;

          QUEUE NAME  :                                  Q1
          NUMBER OF DELETED MESSAGES  :                  000001
```

```
QSTATUS *;
          QUEUE NAME  :                                    Q1
          NUMBER OF COMPLETE MESSAGES  :                   000000
          NUMBER OF MESSAGES IN SEND PHASE  :              000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :           000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :              032767
          NUMBER OF BLOCKS USED FROM POOL  :               000002
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                               DISK

          QUEUE NAME  :                                    Q2
          NUMBER OF COMPLETE MESSAGES  :                   000000
          NUMBER OF MESSAGES IN SEND PHASE  :              000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :           000000
          NUMBER OF BLOCKS ALLOCATED TO THIS QUEUE  :      000040
          NUMBER OF BLOCKS USED FOR THIS QUEUE  :          000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                               CORE

          QUEUE NAME  :                                    Q3
          NUMBER OF COMPLETE MESSAGES  :                   000000
          NUMBER OF MESSAGES IN SEND PHASE  :              000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :           000000
          NUMBER OF BLOCKS ALLOCATED TO THIS QUEUE  :      000040
          NUMBER OF BLOCKS USED FOR THIS QUEUE  :          000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                               CORE

          QUEUE NAME  :                                    Q4
          NUMBER OF COMPLETE MESSAGES  :                   000000
          NUMBER OF MESSAGES IN SEND PHASE  :              000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :           000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :              032767
          NUMBER OF BLOCKS USED FROM POOL  :               000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                               DISK

          QUEUE NAME  :                                    Q5
          NUMBER OF COMPLETE MESSAGES  :                   000000
          NUMBER OF MESSAGES IN SEND PHASE  :              000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :           000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :              032767
          NUMBER OF BLOCKS USED FROM POOL  :               000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                               DISK

          QUEUE NAME  :                                    Q6
          NUMBER OF COMPLETE MESSAGES  :                   000000
          NUMBER OF MESSAGES IN SEND PHASE  :              000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :           000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :              032767
          NUMBER OF BLOCKS USED FROM POOL  :               000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                               DISK
```

```
PURGE Q2 , ALL ;
          QUEUE NAME  :                                    Q2
          NUMBER OF DELETED MESSAGES  :                    000000

QSTATUS Q1 , Q2 ;
          QUEUE NAME  :                                    Q1
          NUMBER OF COMPLETE MESSAGES  :                   000000
          NUMBER OF MESSAGES IN SEND PHASE  :              000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :           000000
          MAXIMUM NUMBER OF BLOCKS IN POOL  :              032767
          NUMBER OF BLOCKS USED FROM POOL  :               000002
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                               DISK

          QUEUE NAME  :                                    Q2
          NUMBER OF COMPLETE MESSAGES  :                   000000
          NUMBER OF MESSAGES IN SEND PHASE  :              000000
          NUMBER OF MESSAGES IN RECEIVE PHASE  :           000000
          NUMBER OF BLOCKS ALLOCATED TO THIS QUEUE  :      000040
          NUMBER OF BLOCKS USED FOR THIS QUEUE  :          000000
          PROGRAM QUEUE
          QUEUE ATTRIBUTES  :                               CORE

PRINT Q1 , Q3 , Q5 , Q6 ;
          QUEUE NAME  :                                    Q1
          NUMBER OF COMPLETE MESSAGES  :                   000000
          QUEUE NAME  :                                    Q3
          NUMBER OF COMPLETE MESSAGES  :                   000000
          QUEUE NAME  :                                    Q5
          NUMBER OF COMPLETE MESSAGES  :                   000000
          QUEUE NAME  :                                    Q6
          NUMBER OF COMPLETE MESSAGES  :                   000000

PURGE Q1 , NUMBMSG = 1 ;
          QUEUE NAME  :                                    Q1
          NUMBER OF DELETED MESSAGES  :                    000000

PURGE Q3 , ALL ;
          QUEUE NAME  :                                    Q3
          NUMBER OF DELETED MESSAGES                       000000

PURGE Q4 , NUMBMSG = 3 ;
          QUEUE NAME  :                                    Q4
          NUMBER OF DELETED MESSAGES  :                    000000

PRINT Q3 ;
          QUEUE NAME  :                                    Q3
          NUMBER OF COMPLETE MESSAGES  :                   000000
```

APPENDIX C

MAM AND QMAINT ERROR MESSAGES

Error messages output after the execution of an MCS application appear in the JOR (job occurrence report).

Error messages output after the execution of the QMAINT utility are,

- those which are written to the SYSOUT file
- those which appear in the JOR

FORMAT OF ERROR MESSAGES

The format of the SYSOUT error message is,

```
ERROR QC nnnn SEVERITY (s) error-message-text
```

The format of the JOR error message is,

```
QCnn error-message-text
```

where,

- QC denotes that the source
  - in the case of the SYSOUT error message is the QMAINT utility
  - in the case of the JOR error message can be either an MCS application or the QMAINT utility
- nnnn and nn, respectively are the number of the message
- s is the severity of the error condition as follows,
  - 2 : warning
  - 3 : fatal, leading to a QMAINT abort but allows a complete syntax analysis
  - 4 : fatal, leading to a QMAINT abort and may prevent syntax analysis
- error-message-text gives the error condition and may be accompanied by the return code of the format

  RC= xxxxxxxx→yyyyyyyy , zzzzzzzz
  - xxxxxxxx : hexadecimal contents of the RC register
  - yyyyyyyy : name of the SIU (system integration unit) procedure
  - zzzzzzzz : return code

MAM JOR
00-05

QC00 BMAM NOT AVAILABLE

syntax : as in text
cause : the step attempted has aborted because
- CNC utility has not yet been run to create the network
- CNC utility is currently executing
- the step itself is multiprocess and was not linked with the option LINKTYPE=BMAM

action : either generate the network or wait until the current CNC session has terminated or link the step, before retrying the job

QC01 MAXIMUM NUMBER OF QASSIGN STATEMENTS IS EXCEEDED

syntax : as in text
cause : the step attempted has aborted due to more than 26 QASSIGN statements in the step enclosure
action : correct the JCL statements and retry the job

QC02 external-queue-name UNKNOWN EXTERNAL QUEUE NAME

syntax : as in text
cause : the step attempted has aborted because a QASSIGN statement specifies an "external-queue-name" that has not yet been defined in the network
action : regenerate the network using CNC utility and retry the job

QC03 external-queue-name QUEUE NOT AVAILABLE

syntax : as in text
cause : the step attempted has been aborted because a QASSIGN statement specifies an "external queue" currently allocated to another step
action : retry the job later when the other step has terminated

QC04 symbolic-queue-name DUPLICATE SYMBOLIC QUEUE NAME

syntax : as in text
cause : the step attempted has aborted because a QASSIGN statement specifies a "symbolic-queue-name" that has already been assigned in another QASSIGN statement of the same step
action : correct the JCL statements and retry the job

QC05 external-queue-name DUPLICATE EXTERNAL QUEUE NAME

syntax : as in text
cause : the step attempted has aborted because a QASSIGN statement specifies an "external-queue-name" that has already been assigned in another QSSIGN statement of the same step
action : correct the JCL statements and retry the job

QC07 BMAM : ABORT USER RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : the contents of RC specifies the system error
cause : the step attempted has aborted due to an error condition at run-time
action : see "Return Codes"

QC08 process-name : UNABLE TO START USER PROCESS

syntax : applicable only to a multiprocess step
cause : the step attempted has aborted because a user process identified by "process-name" cannot be started
action : check the report of the linking process for the load-module and correct the linking as necessary, and retry the job

QC09 UNABLE TO OPEN CTVF FILE RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : the contents of RC specifies the error and system primitive
cause : the system file containing the network description is unable to be opened due to a system error
action : perform ISL with "clean restart", regenerate the network and rerun the job

QC10 CNC SESSION IN PROGRESS

syntax : as in text
cause : the CNC utility is currently executing
action : wait until the current CNC session has terminated before retrying the job.

QC11 QUEUES FILE MEDIA BUSY/NOT AVAILABLE RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : as in text
cause : the disk queue file is not available for processing or has not been mounted
action : retry the job or mount the disk queue file and rerun the job

QC12 MAXIMUM NUMBER OF MAM PROCESS GROUPS EXCEEDED

syntax : as in text
cause : the number of process groups has exceeded the number specified by the MAMNB parameter of the GENCOM command
action : correct the GENCOM command, regenerate the network using CNC utility and retry the job

QC14 external-queue-name : INVALID KEYWORD REPLY IN QASSIGN

syntax : as in text
cause : a QASSIGN statement has a "reply" keyword which does not relate to a program queue
action : correct the JCL statements and retry the job

QC17 keyword : SYNTAX ERROR. INVALID KEYWORD IN CONTEXT

syntax : "keyword" specifies the mismatch between the queue type, either input or output, and the associated parameters in $QASSIGN
cause : see syntax
action : correct the JCL statements, and retry the job.

QC23 external-queue-name ZERO LENGTH QUEUE/SUBQUEUE NAME

syntax : as in text
cause : the step has aborted because the "symbolic-queue-name" of $QASSIGN has been partitioned into "subqueues" of the wrong format, for example, a missing "." or 2 contiguous "."s.
action : correct the format of the "symbolic-queue-name" of the $QASSIGN, and retry the job.

QC24 symbolic-subqueue$_1$-name : "SYMBOLIC SUB-QUEUE-1" FIELD IS BEING FORCED TO SPACES

syntax : as in text
cause : during a RECEIVE, only the "symbolic-queue" represented by either "data-name-1" or QUEUE_NAME is to contain the data.
All other levels of "subqueues" starting with either "data-name-2" or SUBQUEUE_NAME corresponding to "SYMBOLIC SUB-QUEUE-1" are not used and must be set to spaces.
action : correct either the program or the "symbolic-queue-name" in $QASSIGN.

QC25 symbolic-subqueue$_2$-name : "SYMBOLIC SUB-QUEUE-2" FIELD IS BEING FORCED TO SPACES

syntax : as in text
cause : during a RECEIVE, only the "SYMBOLIC QUEUE" and "SYMBOLIC SUB-QUEUE-1" represented by either "data-name-1" and "data-name-2" or QUEUE_NAME and SUBQUEUE_NAME are to contain data.
All other levels of "subqueues" starting with either "data-name-3" or SUBQUEUE2_NAME corresponding to "SYMBOLIC SUB-QUEUE-2" are not used and must be set to spaces.
action : correct either the program or the "symbolic-queue-name" in $QASSIGN.

QC26 symbolic subqueue$_3$-name : "SYMBOLIC SUB-QUEUE-3" FIELD IS BEING FORCED TO SPACES

syntax : as in text
cause : during a RECEIVE, only the "SYMBOLIC QUEUE", "SYMBOLIC SUB-QUEUE-1" and "SYMBOLIC-SUB-QUEUE-2" represented by either "data-name-1", "data-name-2" and "data-name-3" or QUEUE_NAME, SUBQUEUE_NAME and SUBQUEUE2_NAME are to contain data.
The "subqueue" defined by either "data-name-4" or SUBQUEUE3_NAME corresponding to "SYMBOLIC SUB-QUEUE-3" is not used and must be set to spaces.
action : correct either the program or the "symbolic-queue-name" in $QASSIGN.

QC37 QMON ABORTS AT ADDRESS address RC=return-code

syntax : "return-code" is of the format RC=xxxxxxxx→yyyyyyyy , zzzzzzzz
where
- xxxxxxxx : hexadecimal contents of the RC register
- yyyyyyyy : name of the SIU (system integration unit) procedure
- zzzzzzzz : return-code

cause : QMON has aborted because of a system error as indicated by the "return-code"

action : consult the list of general "return-codes" in the Error Messages and Return Codes Reference Manual, and if this message occurs frequently on the job report, call the field engineering service and transmit the return code(s), as applicable.

Note : The network control operator is advised to issue "ST QMON" to restart QMON in order to continue the execution of the MCS application(s).

RETURN CODES
AB - EX

ABTPRC

definition: an invalid internal condition, such as, invalid data or data out of range, has been detected during processing a disk I/O request resulting in discontinuation of file processing
action : take a dump and call the field engineering service

CONFLICT

definition: BTNS has been started for a network generated without LINE definition
action : insert the appropriate LINE command(s) and rerun CNC utility

COUNTOV

definition: the threshold of I/O errors defined at initialization has been exceeded resulting in the non-execution of the pending I/O operation and subsequent shut-down
action :
- either try to copy the file affected and retry the job
- or, if the file fails to be copied, preallocate a new file, reinitialize the system, regenerate the network and retry the job

CPERR

definition: a channel program error or hardware malfunction has occurred
action : retry; if the same condition occurs, call the field engineering service

CPOV

definition: an internal error while trying to start multiple channel programs simultaneously has occurred
action : call the field engineering service

EXTERR

definition: a request has been made to read or write a record outside the limits of the allocated file
action : retry; if the same condition occurs, call the field engineering service

MDNAV

definition: the file was not in the "ready" state when accessed by an I/O operation
action : set the file in the "ready" state either using the same drive or a different drive, and retry the job with the option MAM=YES

MSGOV

definition: a disk I/O request specifying an invalid number of records, such as less than 0 or greater than 5, has been attempted
action : take a dump and call the field engineering service

NEXPDERR

definition: either VCAM or MAM has been preloaded and locked in memory thereby preventing the execution of CNC utility
action : use the CMM command to cancel the locked segment and rerun CNC utility

TABOV

definition: an internal system error has occurred
action : call the field engineering service

The rest of the QC messages apply for the QMAINT utility only.

QC00 BMAM NOT AVAILABLE

syntax : as in text
cause : QMAINT utility has aborted because
- CNC utility has not yet been run to create the network
- CNC utility is currently executing

action : either generate the network or wait until the current CNC session has terminated, before rerunning QMAINT utility

QC07 BMAM : ABORT USER RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : the contents of RC specifies the system error
cause : QMAINT utility has aborted due to an error condition at run-time
action : see "Return Codes"

QC09 UNABLE TO OPEN CTVF FILE RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : the contents of RC specifies the error and system primitive
cause : the system file containing the network description is unable to be opened due to a system error
action : perform ISL with "clean restart", regenerate the network and rerun QMAINT utility

QC10 UNABLE TO ACCESS CTVF FILE RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : the contents of RC specifies the error and system primitive
cause : the system file containing the network description cannot be accessed due to a system error
action : perform ISL with "clean restart", regenerate the network and rerun QMAINT utility

QC11 QUEUES FILE MEDIA BUSY/NOT AVAILABLE RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : as in text
cause : the disk queue file is not available for processing or has not been mounted
action : retry QMAINT utility later or mount the disk queue file and rerun QMAINT utility

QC15 UNABLE TO ACCESS SYSOUT RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : the contents of RC specifies the reason for the abort and the system primitive
cause : QMAINT processing was aborted due to an error in accessing the SYSOUT file
action : retry; if the same condition occurs, call the field engineering service

QMAINT SYSOUT
103 - 302

• ERROR QC | 0103 | SEVERITY (4) ILLEGAL SYNTAX

syntax : ↑ denotes the element in error
cause : one or a combination of the following,
- wrong command name, keyword or argument
- violation of QMAINT reserved syntax

action : correct the command(s) and rerun QMAINT utility

• ERROR QC | 0110 | SEVERITY (4) END OF MESSAGE STRING MISSING

syntax : as in text
cause : a message to be sent to a queue must be terminated in one of the following ways
- either by the ENDMSG option in the SEND command
- or by a card delimiter after the last message text card specifying //EOM starting at column 1

action : correct the message string by either method and rerun QMAINT utility

• ERROR QC | 0113 | SEVERITY (3) MESSAGE SENT TOO LONG

syntax : as in text
cause : an attempt has been made to send a message longer than the maximum acceptable to MAM, resulting in overflow; the maximum is 3053 bytes.
action : truncate the message to be sent and rerun QMAINT utility

• ERROR QC | 0201 | SEVERITY (4) UNABLE TO ACCESS SYSIN
RC= xxxxxxxx→yyyyyyyy , zzzzzzzz

syntax : the contents of RC specifies the reason for the abort
cause : system error
action : check JCL statements and retry the job; if the same condition occurs, call the field engineering service

• ERROR QC | 0302 | SEVERITY (2) INVALID NAME LENGTH : name

syntax : "name" specifies the external queue used in the QMAINT command concerned
cause : the "external-queue-name" must obey the following rules,
- limited to up to 12 alphanumeric characters
- specified previously in an associated QUEUE command

action : correct the "external-queue-name" in the appropriate QMAINT command and rerun QMAINT utility

- ERROR QC | 0304 | SEVERITY (3) DUPLICATE KEYWORD

  syntax : as in text
  cause : a keyword has occurred more than one in a command
  action : correct the appropriate command and rerun QMAINT utility

- ERROR QC | 0306 | SEVERITY (2) INVALID LENGTH PARAMETER

  syntax : as in text
  cause : error in the syntax of the SEND command : the LENGTH value conflicts with the actual length of the message bounded by the string specified in ENDMSG
  action : correct the LENGTH value and rerun QMAINT utility

- ERROR QC | 0307 | SEVERITY (2) QUEUE UNKNOWN : name

  syntax : "name" specifies the external queue used in the QMAINT command concerned
  cause : the "external-queue-name" has not been defined in the network declared
  action : correct the "external-queue-name" in the appropriate QMAINT command and rerun QMAINT utility

- ERROR QC | 0308 | SEVERITY (2) QUEUE NOT AVAILABLE : name

  syntax : "name" specifies the external queue used in the QMAINT command concerned
  cause : the "external-queue-name" identifies either a program queue currently allocated to another application or having active connections, or a terminal-queue to which a terminal is still logged
  action : wait until the queue identified becomes available and rerun QMAINT utility

- ERROR QC | 0405 | SEVERITY (4) FAILED TO CREATE SEGMENT : segment-name
  RC = xxxxxxxx→yyyyyyyy , zzzzzzzz

  syntax : "segment-name" specifies the internal name of a work segment that QMAINT utility was unable to create; the contents of RC specifies the reason for the abort
  cause : system error
  action : retry; if the same condition occurs, call the field engineering service

QMAINT SYSOUT
409 - 412

- ERROR QC | 0409 | SEVERITY (3) ABNORMAL RECEIVE FROM QUEUE : name
  RC = xxxxxxxx→yyyyyyyy , zzzzzzzz

  syntax : "name" specifies the external queue to which a RECEIVE was issued; the contents of RC specifies the reason for the abort
  cause : MCS error
  action : retry; if the same condition occurs, call the field engineering service

- ERROR QC | 0412 | SEVERITY (3) ABNORMAL SEND TO QUEUE : name
  RC = xxxxxxxx→yyyyyyyy , zzzzzzzz

  syntax : "name" specifies the external queue to which a SEND was issued; the contents of RC specifies the reason for the abort
  cause : MCS error
  action : retry; if the same condition occurs, call the field engineering service

APPENDIX D

COMMUNICATIONS STATUS KEY CONDITIONS

The CD entries specify the interface area between MCS and the application, and define the parameters required, as follows,

- between MCS and the application,
  - ACCEPT (H_MSGCNT) : which ascertains the number of messages in a symbolic queue identified by the name of the input CD area of the application
  - RECEIVE (H_RECEIVE) : which requests a message from a specified symbolic queue identified by the name of the input CD area of the application
  - SEND (H_SEND) : which directs a message to a specified symbolic queue identified by the name of the output CD area of the application
- between MCS and the terminals,
  - DISABLE (H_DISABLE) : which terminates the logical connection with specified sources or destinations for data transfers to and/or from the terminals
  - ENABLE (H_ENABLE) : which establishes the logical connection with specified sources or destinations for data transfers to and/or from the terminals.

The CD entries are defined as follows,

- in MCS COBOL, in the Communication Section
- in GPL, by the system primitive H_CD.

The status of this MCS interface is given by a set of status key codes, each uniquely defined by 2 alphanumeric characters denoting the status of each of the parameters, that is, the "x" in the appropriate column of the table denotes the parameter specified.

The parameters listed are in alphabetical order and include all the functions of DISABLE (H_DISABLE) and ENABLE (H_ENABLE).

The status key codes from 9A through 9E are only applicable if the appropriate program-queue on which the application will receive control messages concerning events and the change in terminal status, has been defined with the BREAK option in the QUEUE command at network generation.

COMMUNICATIONS STATUS

Communications Status Key Conditions

* unknown means that symbolic queue is not defined in JCL

Labels and locations are specified in COBOL

Key Code	ACCEPT ($H MSGCNT)	($H)DISABLE INPUT TERMINAL	($H)DISABLE INPUT	($H)DISABLE OUTPUT	($H)ENABLE INPUT TERMINAL	($H)ENABLE INPUT	($H)ENABLE OUTPUT	($H)RECEIVE	($H)SEND	
00	x	x	x	x	x	x	x	x	x	no error detected, action completed
10									x	1 or more destinations disabled, action completed
20	x		x			x		x		1 or more queues/subqueues unknown *, no action taken
		x			x					source unknown *, no action taken
				x			x		x	no action taken for 1 or more destinations unknown *, action taken for known destinations, data-name-4, ERROR KEY, indicates known or unknown *
30				x			x		x	DESTINATION COUNT invalid, no action taken
40		x	x	x	x	x	x			password invalid, no enable/disable action taken
50									x	character count > length of sending field, no action taken
60									x	partial segment with 0 character count or no sending area specified, no action taken
91									x	message data not transferred to queue due to unavailability of mass storage
92								x	x	message data not transferred due to unavailability of memory space
93								x		no data can be input from the terminal to the queue to which a ($H)DISABLE has been issued
94									x	all message data not transferred because maximum message size exceeded, message truncated
95								x		message too long, truncated to maximum size specified
									x	message discarded due to queue allocation overflow

Communications Status Key Conditions
(continued)

Key Code	ACCEPT ($H MSGCNT)	($H)DISABLE INPUT TERMINAL	($H)DISABLE INPUT	($H)DISABLE OUTPUT	($H)ENABLE INPUT TERMINAL	($H)ENABLE INPUT	($H)ENABLE OUTPUT	($H)RECEIVE	($H)SEND	
96								x		message data returned but at least 1 previous message has been lost
97									x	identifier-2 in ($H_)SEND ≠ "0", "1", "2" or "3"
98								x	x	message data not transferred due to I/O error on disk file
99		x	x		x	x		x		access to queue in conflict with JCL definition
9A								x		BREAK has been detected, queue corresponding to symbolic source has been disabled
9B								x		RVI has been detected, queue corresponding to symbolic source has been disabled
9C								x		terminal corresponding to symbolic source has been disconnected
9D								x		terminal corresponding to symbolic source has been connected
9E								x		shutdown is announced, application is required to terminate
9F				x			x		x	access to queue in conflict with JCL definition, or related terminal not logged on to application
9G									x	message not transferred, checkpoint should be taken before attempting further data transfers, applicable to queues with CTLRST option

* unknown means that symbolic queue is not defined in JCL

Labels and locations are specified in COBOL

Bull

Vos remarques sur ce document / Technical publications remarks form

Titre / Title : Data Communications DPS 7 GCOS Communications Processing Facility User's Guide

N° Référence / Reference No. : 47 A2 03UC REV0

Date / Dated : August 1984

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

08 198H - 194 Pages

▶ Vos remarques et suggestions seront attentivement examinées.
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

▶ Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, furnish your complete mailing address below.

NOM/NAME : DATE :

SOCIETE/COMPANY :

ADRESSE/ADDRESS :

■ Remettez cet imprimé à un responsable BULL ou envoyez le directement à :

■ Please give this technical publications remarks form to your BULL representative or mail to :

Bull CEDOC-CELOG
Boîte Postale 110
Parc Industriel d'INCARVILLE